Use of a Quantum Computer to do Importance and Metropolis-Hastings Sampling of a Classical Bayesian Network

Robert R. Tucci
P.O. Box 226
Bedford, MA 01730
tucci@ar-tiste.com

November 12, 2008

Abstract

Importance sampling and Metropolis-Hastings sampling (of which Gibbs sampling is a special case) are two methods commonly used to sample multi-variate probability distributions (that is, Bayesian networks). Heretofore, the sampling of Bayesian networks has been done on a conventional "classical computer". In this paper, we propose methods for doing importance sampling and Metropolis-Hastings sampling of a classical Bayesian network on a quantum computer.

1 Introduction

Monte Carlo methods are frequently used to sample probability distributions. For a single random variable, it is common to draw samples using the inverse transform method[1] or the ARM (acceptance-rejection method)[2]. For an n-tuple of dependent random variables (i.e., a Bayesian network), it is common to use importance sampling (see Appendix A), Gibbs sampling (see Appendix B.1) and Metropolis-Hastings sampling (see Appendix B.2). Two special cases of importance sampling are rejection sampling and likelihood weighted sampling (a.k.a. likelihood weighting).

In previous papers written by me, I define some nets that describe quantum phenomena. I call them "quantum Bayesian nets"(QB nets). They are a counter-

part to the conventional "classical Bayesian nets" (CB nets)[3] that describe classical phenomena.

Heretofore, the sampling of CB nets has been done on a conventional "classical computer". In this paper, we advocate sampling a CB net with a quantum computer.

In Ref.[4], we proposed a method for "embedding" a CB net within a QB net. By applying this embedding technique, we were able to obtain in Ref.[5] a method of doing both rejection sampling and likelihood weighted sampling of a CB net on a quantum computer. In Ref.[5], we illustrated our technique by applying it to a special CB net used in medical diagnosis, the QMR (Quick Medical Reference) CB net.

In this paper, we generalize the results of Ref.[5] to include all kinds of importance sampling, (not just rejection and likelihood weighted sampling). We also show how to do Gibbs sampling and Metropolis-Hastings sampling of a CB net with a quantum computer.

Other workers[6, 7, 8, 9] have considered sampling of a probability distribution using a quantum computer. Their methods are very different from ours. Contrary to them, we utilize a general technique, first proposed in Ref.[4], for embedding CB nets within QB nets. We leave to future work a deeper, more detailed comparison between their methods and ours.

2 Notation and Preliminaries

In this section, we will define some notation that is used throughout this paper. For additional information about our notation, we recommend that the reader consult Ref.[10]. Ref.[10] is a review article, written by the author of this paper, which uses the same notation as this paper.

We will often use the symbol N_B for the number (≥ 1) of qubits and $N_S = 2^{N_B}$ for the number of states with N_B qubits. The quantum computing literature often uses n for N_B and N for N_S, but we will avoid this notation. We prefer to use n for the number operator, defined below.

Let $Bool = \{0, 1\}$. As usual, let $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ represent the set of integers (negative and non-negative), real numbers, and complex numbers, respectively. For integers a, b such that $a \leq b$, let $Z_{a,b} = \{a, a+1, \ldots b-1, b\}$. For any positive integer k and any set S, let S^k denote the Cartesian product of k copies of S; i.e., the set of all k-tuples of elements of S. For any set S, let $|S|$ be the number of elements in S.

We will use $\Theta(S)$ to represent the "truth function"; $\Theta(S)$ equals 1 if statement S is true and 0 if S is false. For example, the Kronecker delta function is defined by $\delta_x^y = \delta(x, y) = \Theta(x = y)$.

Let $\overline{0} = 1$ and $\overline{1} = 0$. If $\vec{a} = a_{N_B-1} \ldots a_2 a_1 a_0$, where $a_\mu \in Bool$, then $dec(\vec{a}) = \sum_{\mu=0}^{N_B-1} 2^\mu a_\mu = a$. Conversely, $\vec{a} = bin(a)$.

We define the single-qubit states $|0\rangle$ and $|1\rangle$ by

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ , \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ . \tag{1}$$

If $\vec{a} \in Bool^{N_B}$, we define the N_B-qubit state $|\vec{a}\rangle$ as the following tensor product

$$|\vec{a}\rangle = |a_{N_B-1}\rangle \otimes \ldots |a_1\rangle \otimes |a_0\rangle \ . \tag{2}$$

For example,

$$|01\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \ . \tag{3}$$

When we write a matrix, and leave some of its entries blank, those blank entries should be interpreted as zeros.

I_k and 0_k will represent the $k \times k$ unit and zero matrices, respectively. For any matrix $A \in \mathbb{C}^{p \times q}$, A^* will stand for its complex conjugate, A^T for its transpose, and $A^\dagger$ for its Hermitian conjugate.

For any matrix A and positive integer k, let

$$A^{\otimes k} = \underbrace{A \otimes \cdots \otimes A \otimes A}_{k \text{ copies of } A} , \tag{4}$$

$$A^{\oplus k} = \underbrace{A \oplus \cdots \oplus A \oplus A}_{k \text{ copies of } A} \ . \tag{5}$$

Suppose $\beta \in Z_{0,N_B-1}$ and M is any 2×2 matrix. We define $M(\beta)$ by

$$M(\beta) = I_2 \otimes \cdots \otimes I_2 \otimes M \otimes I_2 \otimes \cdots \otimes I_2 \ , \tag{6}$$

where the matrix M on the right hand side is located at qubit position β in the tensor product of N_B 2×2 matrices. The numbers that label qubit positions in the tensor product increase from right to left ($\leftarrow$), and the rightmost qubit is taken to be at position 0.

The Pauli matrices are

$$\sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \ , \ \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \ , \ \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ . \tag{7}$$

Let $\vec{\sigma} = (\sigma_X, \sigma_Y, \sigma_Z)$. For any $\vec{a} \in \mathbb{R}^3$, let $\sigma_{\vec{a}} = \vec{\sigma} \cdot \vec{a}$.

The one-qubit Hadamard matrix H is defined as:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \ . \tag{8}$$

The N_B-qubit Hadamard matrix is defined as $H^{\otimes N_B}$.

The number operator n for a single qubit is defined by

$$n = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1 - \sigma_Z}{2} \ . \tag{9}$$

Note that

$$n\,|0\rangle = 0\,|0\rangle = 0 \ \ , \ \ n\,|1\rangle = 1\,|1\rangle \ . \tag{10}$$

We will often use $\overline{n}$ as shorthand for

$$\overline{n} = 1 - n = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \frac{1 + \sigma_Z}{2} \ . \tag{11}$$

Define P_0 and P_1 by

$$P_0 = \overline{n} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = |0\rangle\,\langle 0| \ \ , \ \ P_1 = n = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = |1\rangle\,\langle 1| \ . \tag{12}$$

P_0 and P_1 are orthogonal projection operators and they add to one:

$$P_a P_b = \delta(a,b) P_b \qquad \text{for } \ a, b \in Bool \ , \tag{13}$$

$$P_0 + P_1 = I_2 \ . \tag{14}$$

For $\vec{a} \in Bool^{N_B}$, let

$$P_{\vec{a}} = P_{a_{N_B - 1}} \otimes \cdots \otimes P_{a_2} \otimes P_{a_1} \otimes P_{a_0} \ . \tag{15}$$

For example, with 2 qubits we have

$$P_{00} = P_0 \otimes P_0 = diag(1,0,0,0) \ , \tag{16}$$

$$P_{01} = P_0 \otimes P_1 = diag(0,1,0,0) \ , \tag{17}$$

$$P_{10} = P_1 \otimes P_0 = diag(0,0,1,0) \ , \tag{18}$$

$$P_{11} = P_1 \otimes P_1 = diag(0,0,0,1) \ . \tag{19}$$

Note that

$$P_{\vec{a}} P_{\vec{b}} = \delta(\vec{a}, \vec{b}) P_{\vec{b}} \qquad \text{for } \ \vec{a}, \vec{b} \in Bool^{N_B} \ , \tag{20}$$

$$\sum_{\vec{a} \in Bool^{N_B}} P_{\vec{a}} = I_2 \otimes I_2 \otimes \cdots \otimes I_2 = I_{2^{N_B}} \ . \tag{21}$$

Next we explain our circuit diagram notation. We label single qubits (or qubit positions) by a Greek letter or by an integer. When we use integers, the topmost qubit wire is 0, the next one down is 1, then 2, etc. *Note that in our circuit diagrams, time flows from the right to the left of the diagram.* Careful: Many workers in Quantum Computing draw their diagrams so that time flows from left to right. We eschew their convention because it forces one to reverse the order of the operators every time one wishes to convert between a circuit diagram and its algebraic equivalent in Dirac notation.

Suppose $U \in U(2)$. If τ and κ are two different qubit positions, gate $U(\tau)^{n(\kappa)}$ (or $U(\tau)^{\overline{n}(\kappa)}$) is called a **controlled** U with target τ and control κ. When $U = \sigma_X$, this reduces to a **CNOT (controlled NOT)**. If τ,κ_1 and κ_0 are 3 different qubit positions, $\sigma_X(\tau)^{n(\kappa_1)n(\kappa_0)}$ is called a **Toffoli gate** with target τ and controls κ_1, κ_0. Suppose $N_K \geq 2$ is an integer and $\vec{b} \in Bool^{N_K}$. Suppose $\tau, \kappa_{N_K-1}, \kappa_{N_K-2}, \ldots, \kappa_1, \kappa_0$ are distinct qubits and $\vec{\kappa} = (\kappa_{N_K-1}, \kappa_{N_K-2}, \ldots, \kappa_1, \kappa_0)$. Gate $U(\tau)^{P_{\vec{b}}(\vec{\kappa})}$ is called a **multiply controlled** U with target τ and N_K controls $\vec{\kappa}$. When $U = \sigma_X$, this reduces to an **MCNOT (multiply controlled NOT)**.

For any set Ω and any function $f : \Omega \to \mathbb{C}$, we will use $f(x)/(\sum_{x\in\Omega} num)$, where "num" stands for numerator, to mean $f(x)/(\sum_{x\in\Omega} f(x))$. This notation is convenient when $f(x)$ is a long expression that we do not wish to write twice.

Consider an n-tuple $\vec{f} = (f_1, f_2, \ldots, f_n)$, and a set $A \subset Z_{1,n}$. By $(\vec{f})_A$ we will mean $(f_i)_{i\in A}$; that is, the $|A|$-tuple that one creates from $\vec{f}$, by keeping only the components listed in A.

Symbols which represent random variables will be underlined. The set of values (or states) that a random variable $\underline{x}$ can assume will be denoted by $val(\underline{x})$ (or $S_{\underline{x}}$). Samples of $\underline{x}$ will be denoted by $x^{(k)}$ for $k \in Z_{1,N_{sam}}$.

Next, consider a CB net with nodes $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{N_{nds}}$.

We will use $pa(i)$ ($ch(i)$, respectively) to denote the set of all $j \in Z_{1,N_{nds}}$ such that $\underline{x}_j$ is a parent (child, respectively) of $\underline{x}_i$. Suppose $\gamma = pa, ch$. Let $\gamma(\underline{x}_i) = \{\underline{x}_j : j \in \gamma(i)\}$. Let $\gamma(S) = \cup_{i\in S}\gamma(i)$.

The **Markov blanket** of $\underline{x}_i$ is defined by

$$MB(i) = pa(i) \cup ch(i) \cup pa(ch(i)) \ . \tag{22}$$

Let $\{i\}^c = Z_{1,N_{nds}} - \{i\}$. One an prove that

$$P(x_i|(x)_{\{i\}^c}) = P(x_i|(x)_{MB(i)}) \ . \tag{23}$$

We won't prove Eq.(23) here, but next we will give an example to make it plausible. For the CB net shown in Fig.1, one has

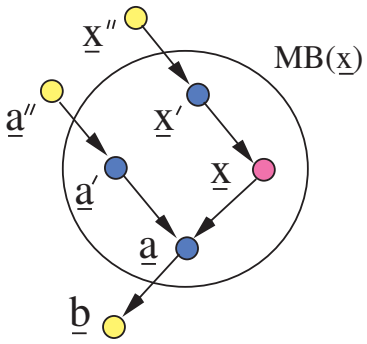


Figure 1: The Markov blanket of node $\underline{x}$ is the set of all nodes inside the large circle, excluding $\underline{x}$.

$$\begin{aligned} P(x|x',x'',a,a',a'',b) &= \frac{P(x|x')P(x'|x'')P(x'')P(a|a',x)P(a'|a'')P(a'')P(b|a)}{\sum_x num} \\ &= \frac{P(x|x')P(a|a',x)}{\sum_x num} \\ &= P(x|x',a,a') \ . \end{aligned} \tag{24}$$

3 Multiplexors

In this section, we discuss some multi-qubit transformations called multiplexors.

Suppose that U is an $N \times N$ unitary matrix, where N is an even number. The **Cosine-Sine Decomposition (CSD) Theorem**[11] states[1] that one can always express U in the form

$$U = \begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} D \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix} , \tag{25a}$$

where the left and right matrices L_0, L_1, R_0, R_1 are $\frac{N}{2} \times \frac{N}{2}$ unitary matrices, and

$$D = \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix} , \tag{25b}$$

$$D_{00} = D_{11} = diag(C_1, C_2, \ldots, C_{\frac{N}{2}}) \ , \tag{25c}$$

$$D_{01} = diag(S_1, S_2, \ldots, S_{\frac{N}{2}}) \ , \quad D_{10} = -D_{01} \ . \tag{25d}$$

[1]Actually, this is only a special case of the CSD Theorem—the case which is most relevant to quantum computing. The general version of the CSD Theorem does not restrict the dimension of U to be even, or even restrict the blocks into which U is partitioned to be of equal size.

For all $i \in Z_{1,\frac{N}{2}}$, $C_i = \cos\theta_i$ and $S_i = \sin\theta_i$ for some angle θ_i. Eqs.(25) can be expressed more succinctly as

$$U = (L_0 \oplus L_1) e^{i\sigma_Y \otimes \Theta} (R_0 \oplus R_1) \ , \tag{26}$$

where $\Theta = diag(\theta_1, \theta_2, \ldots, \theta_{\frac{N}{2}})$.

We will henceforth refer to Ref.[12] as Tuc99. Tuc99 was the first paper to use the CSD to compile unitary matrices. By "compiling a unitary matrix", we mean decomposing it into a SEO (Sequence of Elementary Operators), elementary operators such as single-qubit rotations and CNOTs.

Note that for some $\phi_{\vec{b}} \in \mathbb{R}$ and $N = 2^{N_B}$, matrix D of Eq.(25) can be expressed as

$$\begin{aligned} D &= \exp\left(i\sigma_Y \otimes \sum_{\vec{b} \in Bool^{N_B-1}} \phi_{\vec{b}} P_{\vec{b}} \right) && \text{(27a)} \\ &= \sum_{\vec{b} \in Bool^{N_B-1}} e^{i\phi_{\vec{b}}\sigma_Y} \otimes P_{\vec{b}} && \text{(27b)} \\ &= \prod_{\vec{b} \in Bool^{N_B-1}} e^{i\phi_{\vec{b}}\sigma_Y \otimes P_{\vec{b}}} \ . && \text{(27c)} \end{aligned}$$

To prove that Eqs.(27a), (27b), and (27c) are equivalent, just apply $\left|\vec{b}\right\rangle_{N_B-2,\ldots,1,0}$ with $\vec{b} \in Bool^{N_B-1}$ to the right hand side of each line, and use the fact that $P_{\vec{b}'} \left|\vec{b}\right\rangle = \delta^{\vec{b}'}_{\vec{b}} \left|\vec{b}\right\rangle$. (Note that we can "pull the $\vec{b}$ sum" out of the argument of the exponential only if we also pull out the $\otimes P_{\vec{b}}$.)

In Tuc99, I refer to matrices of the form of the D matrix of Eq.(25) simply as "D-matrices". In my papers that followed Tuc99, I've begun calling such matrices "multiplexors".[2] When I want to be more precise, I call the D matrix of Eq.(25), an $R_y(2)$-multiplexor with target qubit N_B-1 and control qubits $N_B-2, \ldots, 2, 1, 0$. The $R_y(2)$ term refers to the fact that the set of operations acting on the target qubit are 2×2 qubit rotations $R_y(\phi) = e^{i\phi\sigma_Y}$ for some $\phi \in \mathbb{R}$. More generally, one can speak of $U(N)$-multiplexors. Henceforth in this paper, I'll continue using this multiplexor nomenclature, even though it's not used in Tuc99.

Tuc99 gives identities for decomposing an arbitrary $R_y(2)$-multiplexor with $N_B - 1$ controls into a SEO with 2^{N_B-1} CNOTs. Fig.2 shows an example of the SEO decomposition found in Tuc99 for an $R_y(2)$-multiplexor. In Fig.2, 0,1,2,3 are the control qubits, and 4 is the target qubit. The empty square vertices represent

[2] "multiplexor" means "multi-fold" in Latin. A special type of electronic device is also called a multiplexor or multiplexer.

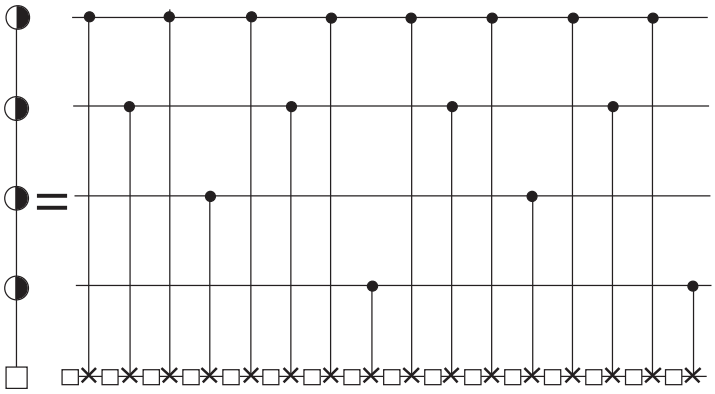


Figure 2: A possible decomposition of an $R_y(2)$-multiplexor with 4 controls.

$R_y(2)$ gates. The symbol to the left of the equal sign, the one with the "half-moon" vertices, was invented by the authors of Ref.[13] to represent a multiplexor.

$U(N)$ multiplexors for any $N \geq 2$ satisfy certain simple properties that we shall discuss next.

Let "con" stand for control and "tar" for target.

Suppose $\vec{b} \in Bool^{N_B,con}$ and $\{U_{\vec{b}}\}_{\forall \vec{b}}$ is a family of $2^{N_{B,tar}} \times 2^{N_{B,tar}}$ unitary matrices. Define

$$E_{\vec{b}}[U_{\vec{b}}] = \sum_{\vec{b}} U_{\vec{b}} \otimes P_{\vec{b}} \, . \tag{28}$$

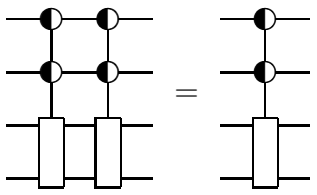
Suppose $\vec{b} \in Bool^{N_B,con1}$, $\vec{b}' \in Bool^{N_B,con2}$, and $\{U_{\vec{b},\vec{b}'}\}_{\forall \vec{b},\vec{b}'}$ is a family of $2^{N_{B,tar}} \times 2^{N_{B,tar}}$ unitary matrices. Define

$$E_{\vec{b}|\vec{b}'}[U_{\vec{b},\vec{b}'}] = \sum_{\vec{b}} e^{U_{\vec{b},\vec{b}'}} \otimes P_{\vec{b}} \, . \tag{29}$$

Claim 1 *If $\vec{b} \in Bool^{N_B,con}$ and $\{U_{\vec{b}}\}_{\forall \vec{b}}$, $\{V_{\vec{b}}\}_{\forall \vec{b}}$ are two families of $2^{N_{B,tar}} \times 2^{N_{B,tar}}$ unitary matrices, then*

$$E_{\vec{b}}[U_{\vec{b}}] \;\; E_{\vec{b}}[V_{\vec{b}}] = E_{\vec{b}}[U_{\vec{b}} V_{\vec{b}}] \, . \tag{30}$$

The last equation can be represented in circuit notation. For example, when $N_{B,con} = 2$ and $N_{B,tar} = 2$, one writes



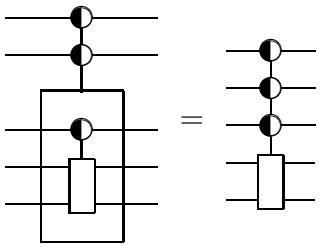
(31)

proof: Obvious.
QED

Claim 2 *If* $\vec{b} \in Bool^{N_B,con1}$, $\vec{b'} \in Bool^{N_B,con2}$, *and* $\{U_{\vec{b},\vec{b'}}\}_{\forall \vec{b},\vec{b'}}$, *is a family of* $2^{N_{B,tar}} \times 2^{N_{B,tar}}$ *unitary matrices, then*

$$E_{\vec{b'}}\left[E_{\vec{b}|\vec{b'}}[U_{\vec{b},\vec{b'}}]\right] = E_{\vec{b},\vec{b'}}[U_{\vec{b},\vec{b'}}] \ . \tag{32}$$

The last equation can be represented in circuit notation. For example, when $N_{B,con1} = 1$, $N_{B,con2} = 2$, *and* $N_{B,tar} = 2$, *one writes*



(33)

proof: Obvious.
QED

We end this section by proving a "chain rule" for $R_y(2)$ multiplexors (similar to the Chapman-Kolgomorov chain rule for conditional probabilities). A result similar to the next claim is given in Ref.[13].

Below, when an index is replaced by a dot, we mean that the index is summed over all its possible values. For example, $q_{\cdot\cdot k} = \sum_{i,j} q_{ijk}$.

Claim 3 *Suppose* $\vec{b} = (b_{N_B-1}, \ldots, b_1, b_0) \in Bool^{N_B}$, $q_{\vec{b}} \geq 0$, $\sum_{\vec{b}} q_{\vec{b}} = 1$ *Assume* $N_B = 3$ *for definiteness. One has*

$$\sum_{\vec{b} \in Bool^3} \sqrt{q_{\vec{b}}} \left|\vec{b}\right\rangle = U \left|0\right\rangle^{\otimes 3} \tag{34a}$$

if

$$U = \left[e^{i\sum_{b_1,b_0} \theta_{b_1 b_0} \sigma_Y \otimes P_{b_1 b_0}}\right] \left[I_2 \otimes e^{i\sum_{b_0} \theta_{b_0} \sigma_Y \otimes P_{b_0}}\right] \left[I_2^{\otimes 2} \otimes e^{i\theta\sigma_Y}\right] \ , \tag{34b}$$

and the angles $\theta_{b_1 b_0}$, θ_{b_0} *and* θ *are defined by*

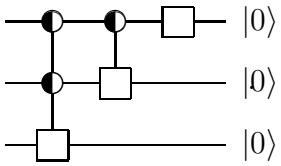
$$(C_{b_1 b_0}, S_{b_1 b_0}) = \frac{1}{\sqrt{q_{\cdot b_1 b_0}}}(\sqrt{q_{0 b_1 b_0}}, \sqrt{q_{1 b_1 b_0}}) \ , \tag{35a}$$

$$(C_{b_0}, S_{b_0}) = \frac{1}{\sqrt{q_{\cdot\cdot b_0}}}(\sqrt{q_{\cdot 0 b_0}}, \sqrt{q_{\cdot 1 b_0}}) \ , \tag{35b}$$

$$(C, S) = (\sqrt{q_{\cdot 0}}, \sqrt{q_{\cdot 1}}) \ , \tag{35c}$$

where $C_\gamma = \cos(\theta_\gamma)$ *and* $S_\gamma = \sin(\theta_\gamma)$ *for any symbol* γ *(including no symbol). Eq.(34) can be represented as a circuit diagram, as follows:*

$$\sum_{\vec{b} \in Bool^3} \sqrt{q_{\vec{b}}} \left|\vec{b}\right\rangle = \tag{36}$$



proof:

$\langle b| P_{b'} = \delta_b^{b'} \langle b|$ for $b, b' \in Bool$. Thus

$$\begin{aligned} \langle b_2, b_1, b_0| U |0\rangle^{\otimes 3} &= \langle b_2| e^{i\sigma_Y \theta_{b_1 b_0}} |0\rangle \langle b_1| e^{i\sigma_Y \theta_{b_0}} |0\rangle \langle b_0| e^{i\sigma_Y \theta} |0\rangle & (37) \\ &= [C^{\overline{b_2}}_{b_1 b_0} S^{b_2}_{b_1 b_0}][C^{\overline{b_1}}_{b_0} S^{b_1}_{b_0}][C^{\overline{b_0}} S^{b_0}] \ . & (38) \end{aligned}$$

Let

$$P(b_2|b_1, b_0) = C^{2\overline{b_2}}_{b_1 b_0} S^{2b_2}_{b_1 b_0} \ , \tag{39a}$$

$$P(b_1|b_0) = C^{2\overline{b_1}}_{b_0} S^{2b_1}_{b_0} \ , \tag{39b}$$

$$P(b_0) = C^{2\overline{b_0}} S^{2b_0} \ . \tag{39c}$$

Then

$$\begin{aligned} q_{\vec{b}} &= \left|\langle b_2, b_1, b_0| U |0\rangle^{\otimes 3}\right|^2 & (40) \\ &= P(b_2|b_1, b_0) P(b_1|b_0) P(b_0) & (41) \\ &= P(\vec{b}) \ . & (42) \end{aligned}$$

Eqs.(39) are satisfied if

$$\frac{\sqrt{q_{b_2 b_1 b_0}}}{\sqrt{q_{\cdot b_1 b_0}}} = C^{\overline{b_2}}_{b_1 b_0} S^{b_2}_{b_1 b_0} \ , \tag{43a}$$

$$\frac{\sqrt{q_{\cdot b_1 b_0}}}{\sqrt{q_{\cdot\cdot b_0}}} = C^{\overline{b_1}}_{b_0} S^{b_1}_{b_0} \ , \tag{43b}$$

$$\sqrt{q_{\cdot\cdot b_0}} = C^{\overline{b_0}} S^{b_0} \ . \tag{43c}$$

QED

The above claim can be easily generalized to arbitrary $N_B > 0$.

Note that $\sum_{\vec{b}} \sqrt{q_{\vec{b}}} \left|\vec{b}\right\rangle$, when expressed in matrix notation, is the column vector with entries $\sqrt{q_{\vec{b}}}$. For example, for $N_B = 3$, it equals $[\sqrt{q_{000}}, \sqrt{q_{001}}, \sqrt{q_{010}}, \ldots, \sqrt{q_{111}}]^T$.

4 Q-Embeddings

In this section, we will review and extend the section of Ref.[4] entitled "Q-Embeddings".

A **probability matrix** $P(y|x)$ is a rectangular (not necessarily square) matrix with row index $y \in S_{\underline{y}}$ and column index $x \in S_{\underline{x}}$ such that $P(y|x) \geq 0$ for all x, y, and $\sum_y P(y|x) = 1$ for all x. A probability matrix is assigned to each node of a CB net.

A unitary matrix $A(y, \tilde{x}|x, \tilde{y})$ (with rows labelled by $y, \tilde{x}$ and columns by $x, \tilde{y}$) is a **q-embedding of probability matrix** $P(y|x)$ if

$$\sum_{\tilde{x}} |A(y, \tilde{x}|x, \tilde{y} = 0)|^2 = P(y|x) \tag{44}$$

for all possible values of y and x. (the "q" in "q-embedding" stands for "quantum"). When considering a q-embedding $A(y, \tilde{x}|x, \tilde{y})$ of a probability matrix, we will refer to y as the **focus index**, $\tilde{y}$ as the **focus-image index** or **source index**, x as the **parent index**, and $\tilde{x}$ as the **parent-image index** or **sink index**. We will also refer to $\tilde{x}$ and $\tilde{y}$ collectively as **ancilla indices**.

Given a QB net $\mathcal{N}^Q$, let

$$P[(x)_L] = \left| \sum_{(x)_{\Gamma_Q - L}} A(x) \right|^2 . \tag{45}$$

On the right hand side of Eq.(45), $A(x)$ is the amplitude of story x, Γ_Q is the set of indices of all the nodes of $\mathcal{N}^Q$, and L is the set of indices of all leaf (a.k.a. external) nodes of $\mathcal{N}^Q$. We say $\mathcal{N}^Q$ is a **q-embedding of CB net** $\mathcal{N}^C$ if $P[(x)_L]$ defined by Eq.(45) satisfies

$$P[(x)_{\Gamma_C}] = \sum_{L_1} P[(x)_L] , \tag{46}$$

where $L_1 \subset L$, and Γ_C is the set of indices of all nodes of $\mathcal{N}^C$. Thus, the probability distribution associated with all nodes of $\mathcal{N}^C$ can be obtained from the probability distribution associated with the external nodes of $\mathcal{N}^Q$. Ref.[4] gives two examples of q-embeddings of CB nets: the two-body scattering net and the Asia net. Ref.[5] gives the example of the Quick Medical Reference net. More examples will be given later in this paper.

4.1 Q-Embeddings of Probability Matrices

In Ref.[4], we showed that any probability matrix has a q-embedding. Our proof was constructive and relied on the Gram-Schmidt method. In this section, we will give a new proof, again constructive, that relies on multiplexors. The q-embeddings constructed in this section, compared with those of Ref.[4], have the advantage that

they are already compiled, whereas those obtained in Ref.[4] in general require a computer program, "a compiler", to compile them numerically.

Eq.(44) is satisfied by

$$A(y,\tilde{x}|\tilde{y}=0,x)=\sqrt{P(y|x)}\ \ \delta_x^{\tilde{x}}\ . \tag{47}$$

When speaking of a parent index $\underline{x}$ and a focus index $\underline{y}$, we will denote number of parent bits by $N_{B,par}=\log_2(N_{\underline{x}})$, and number of focus bits by $N_{B,foc}=\log_2(N_{\underline{y}})$, Also, $N_{S,par}=2^{N_{B,par}}=N_{\underline{x}}$, and $N_{S,foc}=2^{N_{B,foc}}=N_{\underline{y}}$. Eq.(47) can be expressed in matrix form as follows:

$$[A(y,\tilde{x}|\tilde{y}=0,x)]=\begin{array}{c|c} & (\tilde{y}=0,x)\rightarrow \\ \hline (y,\tilde{x}) & D^{0,0} \\ \downarrow & D^{1,0} \\ & \cdots \\ & D^{N_{\underline{y}}-1,0} \end{array}\ , \tag{48}$$

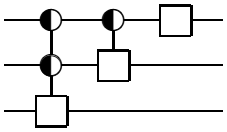
where, for all $y\in val(\underline{y})$, $D^{y,0}\in\mathbb{R}^{N_{\underline{x}}\times N_{\underline{x}}}$ are diagonal matrices with entries

$$(D^{y,0})_{x,\tilde{x}}=\sqrt{P(y|x)}\delta_x^{\tilde{x}}\ . \tag{49}$$

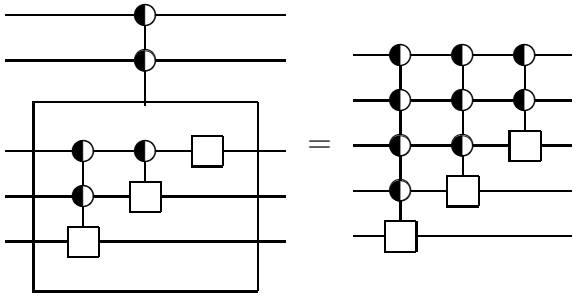
By adding more columns to the matrix of Eq.(48), one can extended it to the following square matrix:

$$\begin{aligned}[A(y,\tilde{x}|\tilde{y},x)] &= \begin{array}{c|cccc} & & (\tilde{y},x) & \rightarrow & \\ \hline (y,\tilde{x}) & D^{0,0} & D^{0,1} & \cdots & D^{0,N_{\underline{y}}-1} \\ \downarrow & D^{1,0} & D^{1,1} & \cdots & D^{1,N_{\underline{y}}-1} \\ & \cdots & \cdots & \cdots & \cdots \\ & D^{N_{\underline{y}}-1,0} & D^{N_{\underline{y}}-1,1} & \cdots & D^{N_{\underline{y}}-1,N_{\underline{y}}-1} \end{array} \qquad (50) \\ &= \sum_{\vec{b}\in Bool^{\log_2 N_{\underline{x}}}} U_{\vec{b}}\otimes P_{\vec{b}}\ . \qquad (51)\end{aligned}$$

For all $y_1,y_2\in val(\underline{y})$, $D^{y_1,y_2}\in\mathbb{R}^{N_{\underline{x}}\times N_{\underline{x}}}$ are diagonal matrices. These diagonal matrices are chosen so that $U_{\vec{b}}\in\mathbb{R}^{N_{\underline{y}}\times N_{\underline{y}}}$ are unitary matrices such that the first column of $U_{\vec{b}}$ is given by $(U_{\vec{b}})_{y,0}=\sqrt{P(y|\underline{x}=dec(\vec{b}))}$. The other columns of the $U_{\vec{b}}$'s can be chosen at will provided that they make the $U_{\vec{b}}$'s unitary. According to Claim 3, we can choose each $U_{\vec{b}}$ to be a chain of multiplexors, in which case $[A(y,\tilde{x}|\tilde{y},x)]$ is a multiplexor of a chain of multiplexors. For example, if $\log_2 N_{\underline{y}}=3$ and $\log_2 N_{\underline{x}}=2$, then

$U_{\vec{b}}=$  , (52)

and

$$[A(y,\tilde{x}|\tilde{y},x)] =$$  . (53)

As mentioned in Section 3 on multiplexors, Ref.[12] shows how to decompose a multiplexor into a SEO. Thus, this particular q-embedding of $P(y|x)$ comes already compiled (decomposed into a SEO).

4.2 Q-Embeddings of CB nets

Ref.[4] describes a method by which, given any CB net $\mathcal{N}^C$, one can construct a QB net $\mathcal{N}^Q$ which is a q-embedding of $\mathcal{N}^C$. In this section, we will review this method. The method will be used in later sections to construct QB nets for sampling.

In the previous section, we showed how to construct a q-embedding for any probability matrix. Now remember that each node of $\mathcal{N}^C$ has a probability matrix assigned to it. The main step in constructing a q-embedding of $\mathcal{N}^C$ is to replace each node matrix of $\mathcal{N}^C$ with a q-embedding of it.

Before describing our construction method, we need some definitions. We say a node $\underline{m}$ is a **marginalizer node** if it has a single input arrow and a single output arrow. Furthermore, the parent node of $\underline{m}$, call it $\underline{x}$, has states $x = (x_1, x_2, \ldots, x_n)$, where $x_i \in S_{\underline{x}_i}$ for each $i \in Z_{1,n}$. Furthermore, for some particular integer $i_0 \in Z_{1,n}$, the set of possible states of $\underline{m}$ is $S_{\underline{m}} = S_{\underline{x}_{i_0}}$, and the node matrix of $\underline{m}$ is $P(\underline{m} = m|\underline{x} = x) = \delta(m, x_{i_0})$.

Let $\mathcal{N}^C$ be a CB net for which we want to obtain a q-embedding. Our construction has two steps, given by Fig.3.

Consider a QB net $\mathcal{N}^Q$ which is a q-embedding of a CB net $\mathcal{N}^C$. In the following sections, we will label some nodes of $\mathcal{N}^Q$ by an underlined group of symbols, such as $\underline{ax}$, followed by an index enclosed in angular brackets, as in $\underline{ax}\langle 4\rangle$. These indices enclosed in angular brackets will be called **worldline indices**. In $\mathcal{N}^Q$, a sequence of random variables such as $\underline{ax}\langle 1\rangle, \underline{ax}\langle 2\rangle, \ldots \underline{ax}\langle n\rangle$, where node $\underline{ax}\langle 1\rangle$ is set to zero and $\underline{ax}\langle n\rangle$ is an external node of $\mathcal{N}^Q$, will be called a **worldline** of ax. When using worldline indices, variables like $(x, \tilde{x}, y, \tilde{y})$ that were used in describing a q-embedding of a probability matrix are replaced by:

$$\begin{array}{l} (\tilde{x}, x) = (x\langle i+1\rangle, x\langle i\rangle) \\ (y, \tilde{y}) = (y\langle j+1\rangle, y\langle j\rangle) \end{array} \quad , \tag{54}$$

for some integers i and j.

(Step 1) Add marginalizer nodes.

More specifically, replace $\mathcal{N}^C$ by a modified CB net ${\mathcal{N}^C}_{mod}$ obtained as follows. For each node $\underline{x}$ of $\mathcal{N}^C$, add a marginalizer node between $\underline{x}$ and every child of $\underline{x}$. If $\underline{x}$ has no children, add a child (with a delta function as probability matrix) to it.

(Step 2) Replace node probability matrices by their q-embeddings. Add ancilla nodes.

More specifically, replace ${\mathcal{N}^C}_{mod}$ by a QB net $\mathcal{N}^Q$ obtained as follows. For each node of ${\mathcal{N}^C}_{mod}$, except for the marginalizer nodes that were added in the previous step, replace its node matrix by a new node matrix which is a q-embedding of the original node matrix. Add a new node for each ancilla index of each new node matrix. These new nodes will be called **ancilla nodes** (of either the source or sink type) because they correspond to ancilla indices.

Figure 3: Algorithm for constructing a q-embedding of a CB net

Every QB net $\mathcal{N}^Q$ can be converted into an equivalent quantum circuit $\mathcal{C}^Q$. To do so, each worldline of $\mathcal{N}^Q$ is turned into the time history of one or more qubits. (The number of qubits in a worldline is log_2 of the number of possible states of node $\langle 1 \rangle$ of the worldline.)

In Ref.[4] we consider two CB nets called Two-Body Scattering and Asia. For each of these CB nets $\mathcal{N}^C$, we perform the steps of Fig.3 to construct a QB net $\mathcal{N}^Q$ that is a q-embedding of $\mathcal{N}^C$. Ref.[4] gives $\mathcal{N}^C$ and $\mathcal{N}^Q$, but not an equivalent quantum circuit $\mathcal{C}^Q$. As examples and for the sake of completeness, Appendix C gives quantum circuits for Two-Body Scattering and Asia.

5 Importance Sampling

In this section, we will propose a method for doing importance sampling of a CB net on a *quantum* computer. The traditional method for doing importance sampling of a CB net on a *classical* computer is reviewed in Appendix A.

Consider a CB net whose nodes are labeled in topological order by $(\underline{x}_1, \underline{x}_2, \ldots \underline{x}_{N_{nds}}) \equiv \underline{x}$. Assume that E (evidence set) and H (hypotheses set) are disjoint subsets of $Z_{1,N_{nds}}$, with $Z_{1,N_{nds}} - E \cup H$ not necessarily empty. Let $X^c = Z_{1,N_{nds}} - X$ for any $X \subset Z_{1,N_{nds}}$. Assume that we are given the prior evidence $(x)_E$, and the number of samples N_{sam}

that we intend to collect.

Suppose x' is an arbitrary point in $val(\underline{x})$. (We'll use the unprimed x, as in $(x)_E$, to denote the evidence.) The probability matrices associated with each node of our CB net will be denoted by $P(x'_i|(x')_{pa(i)})$ for each $i \in Z_{1,N_{nds}}$. In addition, we will assume we are given **sampling probability matrices**, associated with each node of our CB net, denoted by $Q(x'_i|(x')_{pa(i)})$ for each $i \in Z_{1,N_{nds}}$. In all cases, these sampling matrices are constrained to satisfy

$$Q(x'_i|(x')_{pa(i)}) = P(x'_i|(x')_{pa(i)}) \ \ \forall i \in E^c \ . \tag{55}$$

Two important special cases of importance sampling are rejection sampling and likelihood weighted sampling. For **rejection sampling (RS)**,

$$Q(x'_i|(x')_{pa(i)}) = P(x'_i|(x')_{pa(i)}) \ \ \forall i \in Z_{1,N_{nds}} \ . \tag{56}$$

Hence, $Q(x') = P(x')$ for rejection sampling. For **likelihood weighted sampling (LWS) (a.k.a. likelihood weighting)**,

$$Q(x'_i|(x')_{pa(i)}) = \begin{cases} P(x'_i|(x')_{pa(i)}) \ \ \forall i \in E^c \\ \delta(x_i, x'_i) \ \ \forall i \in E \end{cases} \ . \tag{57}$$

Hence, $Q(x') = \delta_{(x)_E}^{(x')_E} \prod_{i \in E^c} P(x'_i|(x')_{pa(i)})$ for likelihood weighted sampling.

Under these assumptions, the algorithm for importance sampling of a CB net on a quantum computer is given by Fig.4 (expressed in pseudo-code, pidgin C language). The only difference between the classical algorithm of Fig.9 and the quantum one of Fig.4 is the underlined command. In the quantum case, we use a quantum computer instead of a classical one to generate ${x_i}^{(k)} \sim Q(x_i|(x^{(k)})_{pa(i)})$. To do this, we can find a q-embedding of $Q(x_i|(x^{(k)})_{pa(i)})$. Starting from an a priori known pure state of the parents $\left|(\underline{x})_{pa(i)} = (x^{(k)})_{pa(i)}\right\rangle$, one applies the q-embedding to it, and then finally one measures x_i.

For example, suppose for some i, $x_i \in Bool$ and $(x)_{pa(i)} = (x_1, x_0) \in Bool^2$. Denote x_i by y. Then a q-embedding A of $Q(x_i|(x)_{pa(i)})$ satisfies

$$A(y, \tilde{x}_1, \tilde{x}_0|\tilde{y} = 0, x_1, x_0) = \sqrt{Q(y|x_1, x_0)} \ \ \delta_{x_1}^{\tilde{x}_1} \delta_{x_0}^{\tilde{x}_0} \ . \tag{58}$$

Suppose that the a priori known pure state of the parents is $\left|(\underline{x})_{pa(i)} = (x'_1, x'_0)\right\rangle$. If we indicate non-zero entries by a plus sign,

For all $(x)_H$ $\{W[(x)_H] = 0;\}$
$W_{tot} = 0;$
For samples $k = 1, 2, \ldots, N_{sam}\{$
  $L = 1;$
  For nodes $i = 1, 2, \ldots, N_{nds}\{$
    Use quantum computer to generate $x_i{}^{(k)}$ from $Q(x_i|(x^{(k)})_{pa(i)});$
    //Here, for LWS, $x_i{}^{(k)} == x_i$ when $i \in E$.
    $//pa(i) \subset Z_{1,i-1}$ so $(x^{(k)})_{pa(i)}$ known at this point.
    if $i \in E\{$
      if $x_i{}^{(k)} == x_i\{$
        $L * = \frac{P(x_i|(x^{(k)})_{pa(i)})}{Q(x_i|(x^{(k)})_{pa(i)})};$
        //Here $\frac{P}{Q} = 1$ for RS and $\frac{P}{Q} = P$ for LWS.
      }else{//LWS never enters here
        go to next k;
      }
    }
  }//i loop (nodes)
  $W[(x^{(k)})_H] + = L;$
  $W_{tot} + = L;$
}//k loop (samples)
For all $(x)_H$ $\{P((x)_H|(x)_E) = \frac{W[(x)_H]}{W_{tot}};\}$

Figure 4: Algorithm for importance sampling of CB net on quantum computer.

$$A\,|\tilde{y}=0,x_1',x_0'\rangle \;=\; \begin{array}{c|cccccc|} & (\tilde{y},x_1,x_2)= & & & & & \\ & 000 & 001 & 010 & 011 & \cdots & \\ \hline (y,\tilde{x}_1,\tilde{x}_2)=000 & + & & & & \cdots & \\ 001 & & + & & & \cdots & \\ 010 & & & + & & \cdots & \\ 011 & & & & + & \cdots & \\ 100 & + & & & & \cdots & \\ 101 & & + & & & \cdots & \\ 110 & & & + & & \cdots & \\ 111 & & & & + & \cdots & \\ \hline \end{array} \;|\tilde{y}=0,x_1',x_0'\rangle \quad (59)$$

$$\rightarrow \sum_{\vec{b}\in Bool^2} e^{i\theta_{\vec{b}}\sigma_Y}\otimes P_{\vec{b}}\,|\tilde{y}=0,x_1',x_0'\rangle \quad (60)$$

$$= \; e^{i\theta_{x_1'x_0'}\sigma_Y(2)}\,|\tilde{y}=0,x_1',x_0'\rangle \;\;, \quad (61)$$

for some $\theta_{\vec{b}} \in \mathbb{R}$. Here the right pointing arrow means that the expression at the origin of the arrow can be extended to the expression at the target of the arrow. Note that, according to Eq.(61), it is not necessary to perform all elementary operations that constitute a decomposition of A. We need only perform one single-qubit rotation picked out by the a priori known state of the parents. The final step is to measure $\underline{y}$ on the state of Eq.(61), without measuring $\underline{x}_1$ and $\underline{x}_0$.

If $\underline{x}_i$ or one of its parent nodes has more than two possible states, then (see Section 4.1) we can still represent the q-embedding A as a multiplexor of a chain of multiplexors. This will give for $A\left|\tilde{x}_i = 0, (x^{(k)})_{pa(i)}\right\rangle$ a chain of multiplexors acting on $\left|\tilde{x}_i = 0, (x^{(k)})_{pa(i)}\right\rangle$. The final step is to measure $\underline{x}_i$, without measuring $(\underline{x})_{pa(i)}$.

Note that since nodes $\underline{x}_j$ for $j \in E$ are fixed, we may treat them as if each had only one possible state. This will reduce the size of the probability matrix $P(x_i|(x)_{pa(i)})$, and of its q-embedding A.

6 Markov Chain Monte Carlo

6.1 Gibbs Sampling

In this section, we will propose a method for doing Gibbs sampling of a CB net on a *quantum* computer. The traditional method for doing Gibbs sampling of a CB net on a *classical* computer is reviewed in Appendix B.1.

Consider a Markov chain $\underline{x}^0 \to \underline{x}^1 \to \underline{x}^2 \ldots \to \underline{x}^T$. Let $\underline{x}^t = (\underline{x}^t_1, \underline{x}^t_2, \ldots, \underline{x}^t_{N_{nds}})$ for each time t represent a separate copy of a CB net with nodes $\underline{x}^t_1, \underline{x}^t_2, \ldots, \underline{x}^t_{N_{nds}}$, and probability matrices $P(x^t_i|(x^t)_{pa(i)})$. (The nodes of the CB net $\underline{x}^t$ are not necessarily in topological order.) Assume that E (evidence set) and H (hypotheses set) are disjoint subsets of $Z_{1,N_{nds}}$, with $Z_{1,N_{nds}} - E \cup H$ not necessarily empty. Let $X^c = Z_{1,N_{nds}} - X$ for any $X \subset Z_{1,N_{nds}}$. Assume that we are given the prior evidence $(x)_E$. All probabilities in this section about the Gibbs algorithm will be conditioned implicitly on $(\underline{x}^t)_E = (x)_E$ for all t. The Gibbs algorithm is designed to respect this constraint, by never changing the value of $(\underline{x}^t)_E$ after it is initially set.

Suppose that we wish to sweep through all nodes of graph $\underline{x}^t$, in a fixed deterministic order, repeating this all-nodes-sweep N_{gra} times. t will change by one every time one node is visited. Thus, the last time T of the Markov chain will be $N_{nds}N_{gra}$. Suppose we wish to sweep through β copies of the graph $\underline{x}^t$ before performing each measurement. Assume that we are given the burn time t_{burn} ($0 << t_{burn} << T$).

Under these assumptions, the algorithm for Gibbs sampling of a CB net on a quantum computer is given by Fig.5 (expressed in pseudo-code, pidgin C language). When $\beta = 1$, the only difference between the classical algorithm of Fig.11 and the quantum one of Fig.5 is that the two underlined commands in the quantum algorithm replace the i loop (over nodes) in the classical one. In the quantum case, we use a quantum computer to generate $x^{t+\beta N_{nds}} \sim P(x^{t+\beta N_{nds}}|x^t)$. To do this, we find a CB

For all $(x)_H$ $\{W[(x)_H] = 0;\}$
$W_{tot} = 0$;
Initialize $\underline{x}^0$ to some value x^0, subject to $(x^0)_E = (x)_E$;
$t = 0$;
For graphs $g = \beta, 2\beta, 3\beta, \ldots, N_{gra}\{$
Use quantum computer to generate $x^{t+\beta N_{nds}} \sim P(x^{t+\beta N_{nds}}|x^t)$;
$t+ = \beta N_{nds}$;
if $t > t_{burn}\{$// $0 << t_{burn} << N_{nds}N_{gra}$
$W[(x^t)_H] + +$;
$W_{tot} + +$;
$\}$
$\}$//g loop (graphs)
For all $(x)_H$ $\{P((x)_H|(x)_E) = \frac{W[(x)_H]}{W_{tot}};\}$

Figure 5: Algorithm for Gibbs sampling of CB net on quantum computer. Nodes of CB net $\underline{x}^t$ are visited in a fixed deterministic order.

net that generates $P(x^{t+\beta N_{nds}}|x^t)$ and then we find a q-embedding of that CB net.

As an example, suppose $N_{nds} = 3$ and $\beta = 2$. Fig.6 shows a CB net that will generate $P(x^{t+\beta N_{nds}}|x^t)$ provided that we set

$$P(x_i^t) = \delta(x_i^t, (x_i^t)_{prev}) \tag{62}$$

for $i \in Z_{1,3}$. Here $(x_i^t)_{prev}$ are the values of x_i^t obtained previously in the algorithm.

Note that Fig.6 is divided into 7 "time slices" $t + j$ for $j \in Z_{0,6}$. Eq.(62) gives the probability matrices associated with the nodes of the first time slice. The probability matrices associated with the nodes of the other 6 time slices are as follows. For $j \in Z_{0,5}$,

$$P(x_{1\oplus j}^{t+1+j}|x_{2\oplus j}^{t+j}, x_{3\oplus j}^{t+j}) = P_{\underline{x}_{1\oplus j}|\underline{x}_{2\oplus j},\underline{x}_{3\oplus j}}(x_{1\oplus j}^{t+1+j}|x_{2\oplus j}^{t+j}, x_{3\oplus j}^{t+j}) \ , \tag{63a}$$

$$P(x_{2\oplus j}^{t+1+j}|x_{2\oplus j}^{t+j}) = \delta(x_{2\oplus j}^{t+1+j}, x_{2\oplus j}^{t+j}) \ , \tag{63b}$$

$$P(x_{3\oplus j}^{t+1+j}|x_{3\oplus j}^{t+j}) = \delta(x_{3\oplus j}^{t+1+j}, x_{3\oplus j}^{t+j}) \ , \tag{63c}$$

where $\oplus$ denotes addition mod 3 with 3 and 0 identified.[3] Here $P_{\underline{x}_{1\oplus j}|\underline{x}_{2\oplus j},\underline{x}_{3\oplus j}}$ is a node probability of the CB net $\underline{x}^t$. We can replace $P_{\underline{x}_{1\oplus j}|\underline{x}_{2\oplus j},\underline{x}_{3\oplus j}}$ by $P_{\underline{x}_{1\oplus j}|(\underline{x})_{MB(1\oplus j)}}$

[3] In C language, $\mathtt{x} \oplus \mathtt{y} = ((\mathtt{x}+\mathtt{y})\%\mathtt{3} == \mathtt{0}?\mathtt{3} : (\mathtt{x}+\mathtt{y})\%\mathtt{3})$, where x, y are non-negative integers.

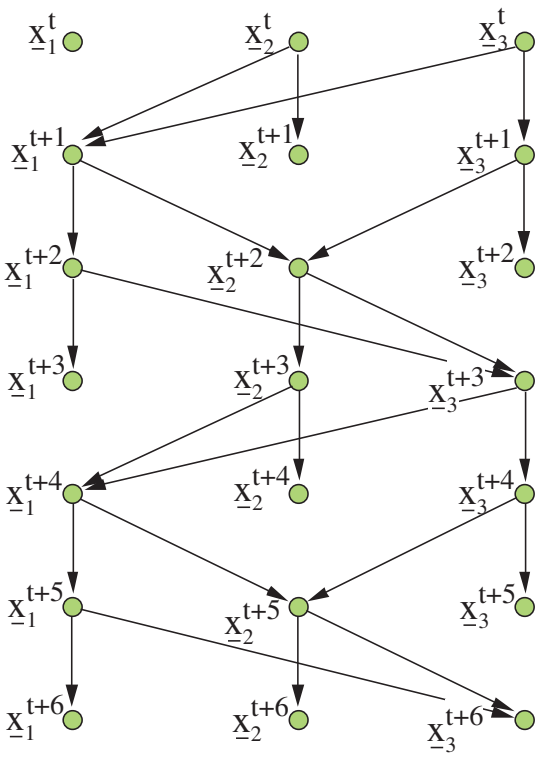


Figure 6: CB net for Gibbs sampling algorithm, for a CB net $\underline{x}^t$ with 3 nodes, sweeping through the nodes of $\underline{x}^t$ in a fixed deterministic order, repeating this all-nodes-sweep twice.

if the Markov blanket of $\underline{x}_{1\oplus j}$ does not include all nodes in $(\underline{x})_{\{1\oplus j\}^c}$. For example, when $j = 0$, we get the conditional probabilities of the nodes of the first time slice:

$$P(x_1^{t+1}|x_2^t, x_3^t) = P_{\underline{x}_1|\underline{x}_2,\underline{x}_3}(x_1^{t+1}|x_2^t, x_3^t) \ , \tag{64a}$$

$$P(x_2^{t+1}|x_2^t) = \delta(x_2^{t+1}, x_2^t) \ , \tag{64b}$$

$$P(x_3^{t+1}|x_3^t) = \delta(x_3^{t+1}, x_3^t) \ . \tag{64c}$$

Here $P_{\underline{x}_1|\underline{x}_2,\underline{x}_3}$ is a node probability of the CB net $\underline{x}^t$. We can replace $P_{\underline{x}_1|\underline{x}_2,\underline{x}_3}$ by $P_{\underline{x}_1|(\underline{x})_{MB(1)}}$ if the Markov blanket of $\underline{x}_1$ does not include all nodes in $(\underline{x})_{\{1\}^c}$.

The full probability distribution for the net of Fig.6 is given by:

$$P(x^t, x^{t+1}, x^{t+2}, x^{t+3}, x^{t+4}, x^{t+5}, x^{t+6}) = \left\{ \begin{array}{ccc} P(x_1^t) & P(x_2^t) & P(x_3^t) \\ & | & | \\ P(x_1^{t+1} & |x_2^{t+1} & , x_3^{t+1}) \\ & \diagdown\!\diagup & \\ P(x_2^{t+2} & |x_3^{t+2} & , x_1^{t+2}) \\ & \diagdown\!\diagup & \\ P(x_3^{t+3} & |x_1^{t+3} & , x_2^{t+3}) \\ & \diagdown\!\diagup & \\ P(x_1^{t+4} & |x_2^{t+4} & , x_3^{t+4}) \\ & \diagdown\!\diagup & \\ P(x_2^{t+5} & |x_3^{t+5} & , x_1^{t+5}) \\ & \diagdown\!\diagup & \\ P(x_3^{t+6} & |x_1^{t+6} & , x_2^{t+6}) \end{array} \right\} , \quad (65)$$

where each of the slanted lines represents a Kronecker delta function equating the two variables at the two ends of the line. Summing the full probability distribution over all nodes except the final ones $\underline{x}^{t+6}$ yields:

$$P(x^{t+6}) = \sum_{x_1^t, x_2^{t+1}, x_3^{t+2}} \quad \sum_{x_1^{t+3}, x_2^{t+4}, x_3^{t+5}} \left\{ \begin{array}{l} P(x_1^t) P(x_2^{t+1}) P(x_3^{t+2}) \\ P(x_1^{t+3} | x_2^{t+1}, x_3^{t+2}) \\ P(x_2^{t+4} | x_3^{t+2}, x_1^{t+3}) \\ P(x_3^{t+5} | x_1^{t+3}, x_2^{t+4}) \\ P(x_1^{t+6} | x_2^{t+4}, x_3^{t+5}) \\ P(x_2^{t+6} | x_3^{t+5}, x_1^{t+6}) \\ P(x_3^{t+6} | x_1^{t+6}, x_2^{t+6}) \end{array} \right\} . \quad (66)$$

It is convenient to make the following change of notation:

$$\begin{array}{l} (x_1^t, x_2^{t+1}, x_3^{t+2}) \rightarrow (X_1^t, X_2^t, X_3^t) \\ (x_1^{t+3}, x_2^{t+4}, x_3^{t+5}) \rightarrow (X_1^{t+3}, X_2^{t+3}, X_3^{t+3}) \\ x^{t+6} \rightarrow X^{t+6} \end{array} \quad (67)$$

Described more succinctly, what we are doing is replacing x by X and $t + j$ by $t+(j/3)3$ for $j \in Z_{0,6}$, where the division by 3 is "integer division", with no remainder, an operation available in most computer languages. In the new notation, Eq.(66) simplifies to

$$P(x^{t+6}) = \sum_{X^t} \quad \sum_{X^{t+3}} \left\{ \begin{array}{l} P(X^t) \\ P(X_1^{t+3}|X_2^t, X_3^t) \\ P(X_2^{t+3}|X_3^t, X_1^{t+3}) \\ P(X_3^{t+3}|X_1^{t+3}, X_2^{t+3}) \\ P(X_1^{t+6}|X_2^{t+3}, X_3^{t+3}) \\ P(X_2^{t+6}|X_3^{t+3}, X_1^{t+6}) \\ P(X_3^{t+6}|X_1^{t+6}, X_2^{t+6}) \end{array} \right\}, \tag{68}$$

or, equivalently,

$$P(x^{t+6}) = \sum_{X^t, X^{t+3}} P(X^t) \prod_{j=1,2} \left\{ \begin{array}{l} P(X_1^{t+3j}|X_2^{t+3(j-1)}, X_3^{t+3(j-1)}) \\ P(X_2^{t+3j}|X_3^{t+3(j-1)}, X_1^{t+3j}) \\ P(X_3^{t+3j}|X_1^{t+3j}, X_2^{t+3j}) \end{array} \right\}. \tag{69}$$

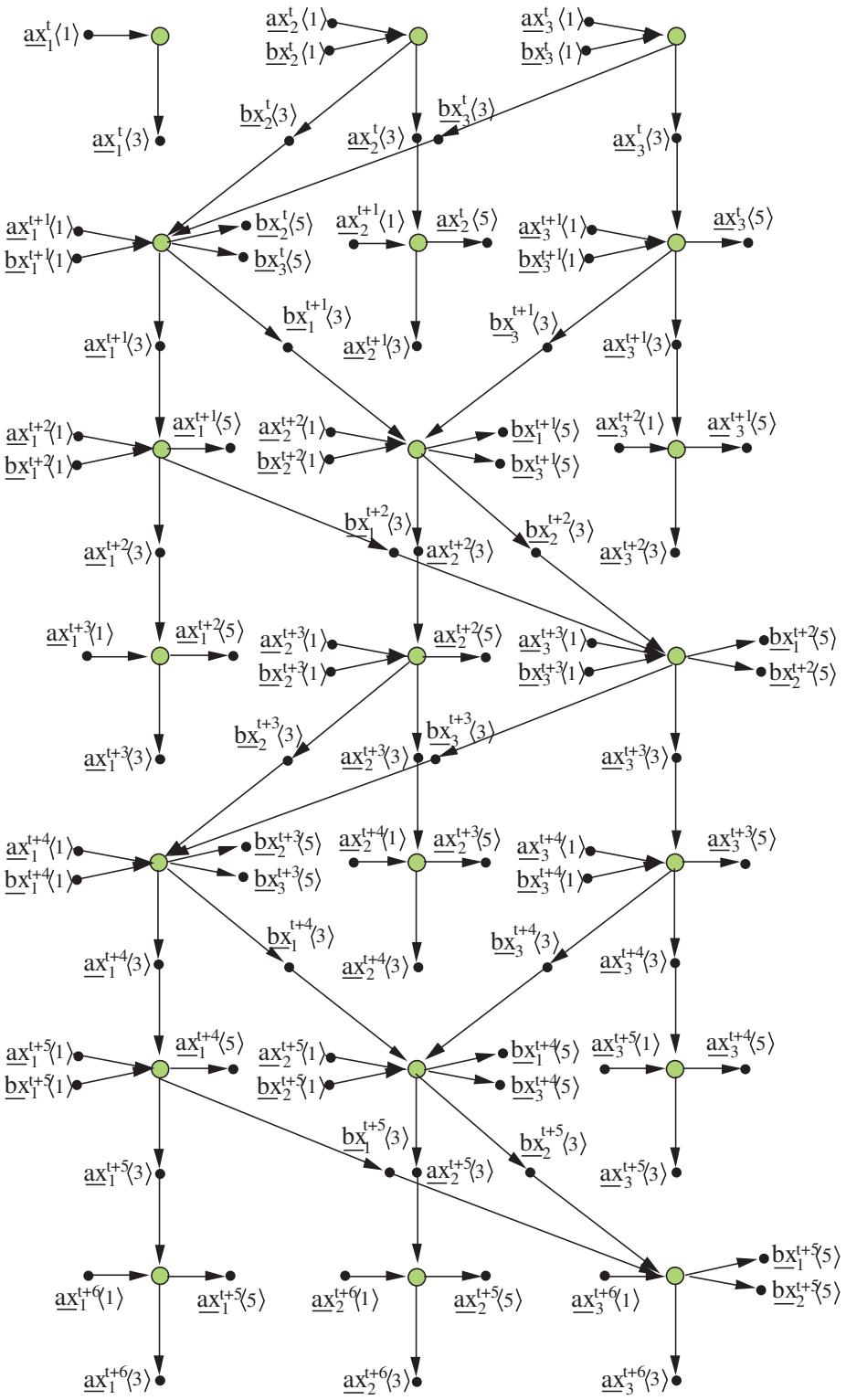


Figure 7: QB net that is a q-embedding of the CB net of Fig.6.

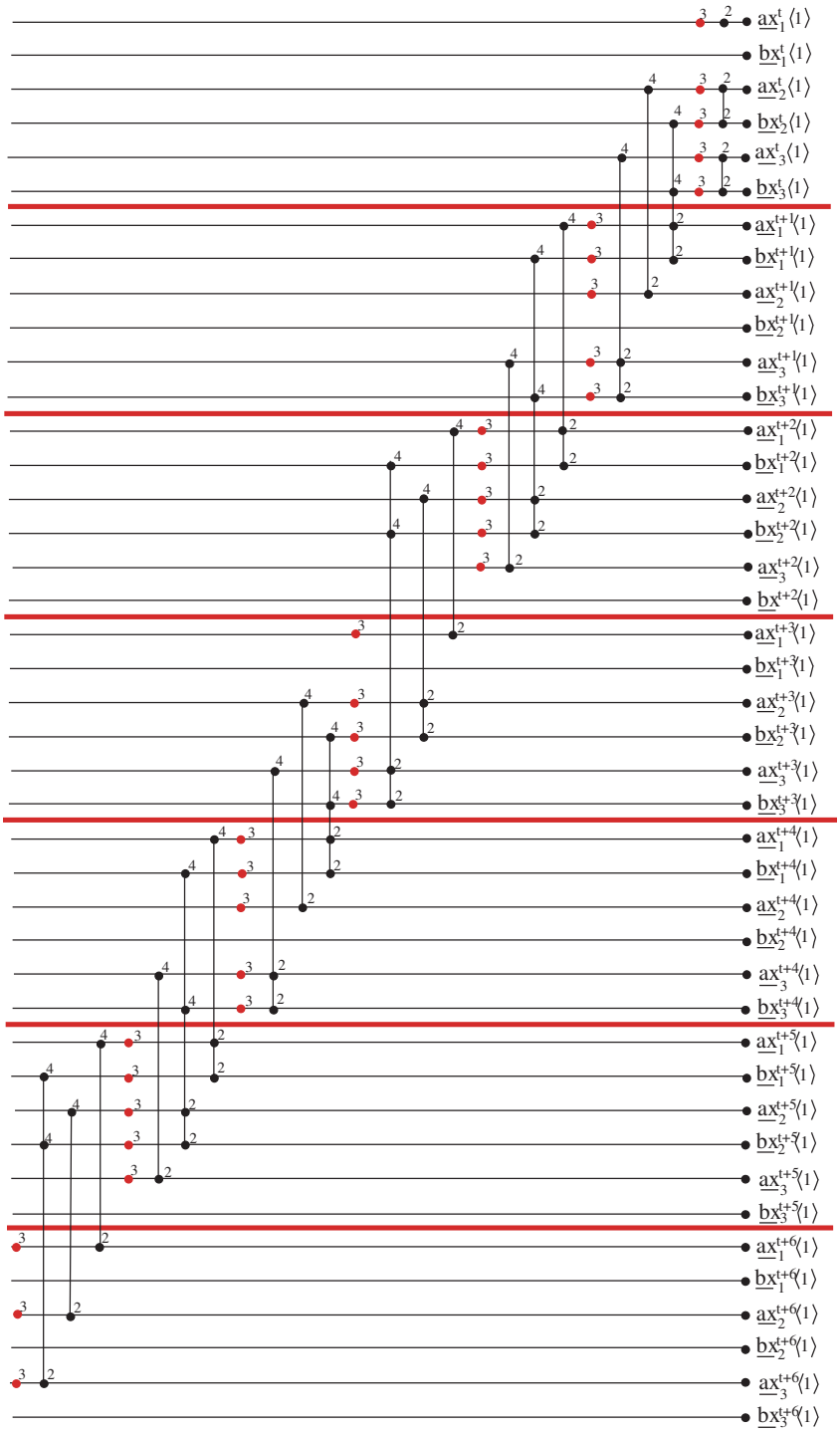


Figure 8: Quantum circuit for QB net of Fig.7

Following the steps of Fig.3, we obtain the QB net Fig.7, a q-embedding of the

CB net of Fig.6. In Fig.7, due to lack of space, labels for the original (light-colored, not black) nodes have been omitted. These omitted labels can be re-constituted as follows. If the original node has incoming arrows $\underline{z}_1, \underline{z}_2, \ldots, \underline{z}_n$, then the node is labeled by the n-tuple $(\underline{z}_1, \underline{z}_2, \ldots, \underline{z}_n)$.

From the QB net of Fig.7, one easily obtains the equivalent quantum circuit shown in Fig.8. Some simple observations about this circuit are:

- The first time slice t uses only 4 qubits, the intermediate ones use 5 qubits, and the final one $t+6$ uses 3 qubits.
- Once the "column of $\langle 3\rangle$'s" is reached midway into time slice $t+j$ for $j \in Z_{1,5}$, the qubits of the previous time slice can be recycled (used again). Also, in each time slice, at least one of the six qubits is never used. Thus, we only need $5+5=10$ qubits for $N_{nds}=3$, or $2(2N_{nds}-1)$ in general.
- All worldlines except those of the first time slice start at $|0\rangle$. In the first time slice, $ax_i^t\langle 1\rangle$ and $bx_i^t\langle 1\rangle$ start at $|(x_i^t)_{prev}\rangle$ for $i=1,2,3$.
- Nodes $\langle 3\rangle$ may be omitted since they do not change the state of their qubit.

Let Γ_{ext} be the set of all external nodes of the QB net of Fig.7, and let Γ_{int} be the set of all other nodes. Let A be the full amplitude of the QB net. Then

$$P(ax^{t+6}\langle 3\rangle|(x^t)_{prev}) = \sum_{\Gamma_{ext}-\{ax^{t+6}\langle 3\rangle\}} \left|\sum_{\Gamma_{int}} A\right|^2 . \tag{70}$$

By construction, the probability $P(ax^{t+6}\langle 3\rangle|(x^t)_{prev})$ and the probability $P(x^{t+6}|(x^t)_{prev})$ of Eq.(69) should be equal, if we equate $ax^{t+6}\langle 3\rangle$ and x^{t+6}.

Next, we will give the node amplitudes for the nodes of quantum circuit Fig.8. We will give node amplitudes only when those amplitudes are non-trivial (i.e., not equal to just a delta function, like they are for the $\langle 1\rangle$ and $\langle 3\rangle$ nodes). We will give compilations for these non-trivial node amplitudes assuming $val(\underline{a}) \in Bool$ for all nodes $\underline{a}$ of the CB net $\underline{x}^t$. If some node has more than two possible values, then, we increase the number of values of that node to a power of two. Compilation of node amplitudes in this case is slightly more complicated than when all nodes have only two possible values, but it can still be done using the multiplexor techniques of Section 4.

Nontrivial nodes in initial time slice t:

1. Nodes $(\underline{ax}_2^t\langle 2\rangle, \underline{bx}_2^t\langle 2\rangle)$ and $(\underline{ax}_3^t\langle 2\rangle, \underline{bx}_3^t\langle 2\rangle)$. These two nodes are analogous. Consider the first one for definiteness. One has

$$\begin{array}{c} A(ax_2^t\langle 2\rangle, bx_2^t\langle 2\rangle|ax_2^t\langle 1\rangle = (x_2^t)_{prev}, bx_2^t\langle 1\rangle = (x_2^t)_{prev}) = \\ \Theta(ax_2^t\langle 2\rangle = ax_2^t\langle 1\rangle = (x_2^t)_{prev})\Theta(bx_2^t\langle 2\rangle = bx_2^t\langle 1\rangle = (x_2^t)_{prev}) \end{array} . \tag{71}$$

One can extend this A to the identity matrix, if it acts on $\left|\underline{ax}_2^t\langle 1\rangle = (x_2^t)_{prev}\right\rangle \left|\underline{bx}_2^t\langle 1\rangle = (x_2^t)_{prev}\right\rangle$.

2. Node $\underline{ax}_1^t\langle 2\rangle$. One has

$$\begin{array}{c} A(ax_1^t\langle 2\rangle | ax_1^t\langle 1\rangle = (x_1^t)_{prev}) = \\ \Theta(ax_1^t\langle 2\rangle = ax_1^t\langle 1\rangle = (x_1^t)_{prev}) \end{array} \ . \tag{72}$$

One can extend this A to the identity matrix, if it acts on $|\underline{ax}_1^t\langle 1\rangle = (x_1^t)_{prev}\rangle$.

Nontrivial nodes straddling time slices $t+j$ and $t+j+1$ for $j \in Z_{0,4}$:

1. Nodes $(\underline{bx}_{f(j)}^{t+1+j}\langle 2\rangle, \underline{ax}_{f(j)}^{t+1+j}\langle 2\rangle, \underline{bx}_{f(j)\oplus 2}^{t+j}\langle 4\rangle, \underline{bx}_{f(j)\oplus 1}^{t+j}\langle 4\rangle)$ where $\begin{array}{c|ccccc} j & 0 & 1 & 2 & 3 & 4 \\ \hline f(j) & 1 & 2 & 3 & 1 & 2 \end{array}$, and the $\oplus$ denotes mod 3 addition with 3 and 0 identified. These five nodes are analogous. Consider $j = 0$ for definiteness.

For node $(\underline{bx}_1^{t+1}\langle 2\rangle, \underline{ax}_1^{t+1}\langle 2\rangle, \underline{bx}_3^t\langle 4\rangle), \underline{bx}_2^t\langle 4\rangle)$, one has

$$\begin{array}{c} A(bx_1^{t+1}\langle 2\rangle, ax_1^{t+1}\langle 2\rangle, bx_3^t\langle 4\rangle, bx_2^t\langle 4\rangle | bx_1^{t+1}\langle 1\rangle = 0, ax_1^{t+1}\langle 1\rangle = 0, bx_3^t\langle 3\rangle, bx_2^t\langle 3\rangle) = \\ \sqrt{P_{\underline{x}_1|\underline{x}_3,\underline{x}_2}(ax_1^{t+1}\langle 2\rangle | bx_3^t\langle 3\rangle, bx_2^t\langle 3\rangle)}\ \delta_{ax_1^{t+1}\langle 2\rangle}^{bx_1^{t+1}\langle 2\rangle} \delta_{bx_3^t\langle 3\rangle}^{bx_3^t\langle 4\rangle} \delta_{bx_2^t\langle 3\rangle}^{bx_2^t\langle 4\rangle} \end{array} \ . \tag{73}$$

If we indicate non-zero entries by a plus sign,

$$A \ = \tag{74}$$

$(bx_1^{t+1}, ax_1^{t+1}, bx_3^t, bx_2^t)=$	0000	0001	0010	0011	$\cdots$
0000	+				$\cdots$
0001		+			$\cdots$
0010			+		$\cdots$
0011				+	$\cdots$
0100					$\cdots$
0101					$\cdots$
0110					$\cdots$
0111					$\cdots$
1001					$\cdots$
1010					$\cdots$
1011					$\cdots$
1100	+				$\cdots$
1101		+			$\cdots$
1110			+		$\cdots$
1111				+	$\cdots$

$$\rightarrow \quad \sigma_X(3)^{n(2)} I_2 \otimes \sum_{\vec{b}\in Bool^2} e^{i\theta_{\vec{b}}\sigma_Y} \otimes P_{\vec{b}} \tag{75}$$

$$= \quad \sigma_X(3)^{n(2)} \sum_{\vec{b}\in Bool^2} e^{i\theta_{\vec{b}}\sigma_Y(2)} P_{\vec{b}}(1,0) \ , \tag{76}$$

for some $\theta_{\vec{b}} \in \mathbb{R}$. This choice of A can be compiled using multiplexor methods.

2. Nodes $(\underline{ax}^{t+1+j}_{f(j)}\langle 2\rangle, \underline{ax}^{t+j}_{f(j)}\langle 4\rangle)$ where $\begin{array}{c|ccccc} j & 0 & 1 & 2 & 3 & 4 \\ \hline f(j) & 2 & 3 & 1 & 2 & 3 \end{array}$. These five nodes are analogous. Consider $j = 0$ for definiteness.

   For node $(\underline{ax}^{t+1}_2\langle 2\rangle, \underline{ax}^{t}_2\langle 4\rangle)$, one has

$$\begin{array}{l} A(ax^{t+1}_2\langle 2\rangle, ax^t_2\langle 4\rangle | ax^{t+1}_2\langle 1\rangle = 0, ax^t_2\langle 3\rangle) = \\ \Theta(ax^{t+1}_2\langle 2\rangle = ax^t_2\langle 4\rangle = ax^t_2\langle 3\rangle) \end{array} . \tag{77}$$

   Thus,

$$A \;=\; \begin{array}{r|r|l|l} & \begin{array}{r}(ax^{t+1}_2\langle 1\rangle, ax^t_2\langle 3\rangle) = \\ 00\end{array} & 01 & \cdots \\ \hline (ax^{t+1}_2\langle 2\rangle, ax^t_2\langle 4\rangle) = 00 & 1 & 0 & \cdots \\ 01 & 0 & 0 & \cdots \\ 10 & 0 & 0 & \cdots \\ 11 & 0 & 1 & \cdots \end{array} \tag{78}$$

$$\rightarrow \;\; \sigma_X(1)^{n(0)} I_2^{\otimes 2} \; . \tag{79}$$

   This operation is unnecessary except at the end, between time slices $t + 5$ and $t + 6$.

3. Nodes $(\underline{bx}^{t+1+j}_{f(j)}\langle 2\rangle, \underline{ax}^{t+1+j}_{f(j)}\langle 2\rangle, \underline{ax}^{t+j}_{f(j)}\langle 4\rangle)$ where $\begin{array}{c|ccccc} j & 0 & 1 & 2 & 3 & 4 \\ \hline f(j) & 3 & 1 & 2 & 3 & 1 \end{array}$. These five nodes are analogous. Consider $j = 0$ for definiteness.

   For node $(\underline{bx}^{t+1}_3\langle 2\rangle, \underline{ax}^{t+1}_3\langle 2\rangle, \underline{ax}^{t}_3\langle 4\rangle)$, one has

$$\begin{array}{l} A(bx^{t+1}_3\langle 2\rangle, ax^{t+1}_3\langle 2\rangle, ax^t_3\langle 4\rangle | bx^{t+1}_3\langle 1\rangle = 0, ax^{t+1}_3\langle 1\rangle = 0, ax^t_3\langle 3\rangle) = \\ \Theta(bx^{t+1}_3\langle 2\rangle = ax^{t+1}_3\langle 2\rangle = ax^t_3\langle 4\rangle = ax^t_3\langle 3\rangle) \end{array} . \tag{80}$$

   Thus,

$$A \;=\; \begin{array}{r|r|l|l} & \begin{array}{r}(bx^{t+1}_3\langle 1\rangle, ax^{t+1}_3\langle 1\rangle, ax^t_3\langle 3\rangle) = \\ 000\end{array} & 001 & \cdots \\ \hline (bx^{t+1}_3\langle 2\rangle, ax^{t+1}_3\langle 2\rangle, ax^t_3\langle 4\rangle) = 000 & 1 & 0 & \cdots \\ 001 & 0 & 0 & \cdots \\ 010 & 0 & 0 & \cdots \\ 011 & 0 & 0 & \cdots \\ 100 & 0 & 0 & \cdots \\ 101 & 0 & 0 & \cdots \\ 110 & 0 & 0 & \cdots \\ 111 & 0 & 1 & \cdots \end{array} \tag{81}$$

$$\rightarrow \;\; [\sigma_X(2)\sigma_X(1)]^{n(0)} I_2^{\otimes 3} \; . \tag{82}$$

Nontrivial nodes straddling time slices $t+5$ and $t+6$:

1. Nodes $(\underline{ax}_1^{t+6}\langle 2\rangle, \underline{ax}_1^{t+5}\langle 4\rangle)$ and $(\underline{ax}_2^{t+6}\langle 2\rangle, \underline{ax}_2^{t+5}\langle 4\rangle)$. These two nodes are analogous. Consider the first one for definiteness. One has

$$\begin{array}{l} A(ax_1^{t+6}\langle 2\rangle, ax_1^{t+5}\langle 4\rangle | ax_1^{t+6}\langle 1\rangle = 0, ax_1^{t+5}\langle 3\rangle) = \\ \Theta(ax_1^{t+6}\langle 2\rangle = ax_1^{t+5}\langle 4\rangle = ax_1^{t+5}\langle 3\rangle) \end{array} \quad . \tag{83}$$

Thus,

$$A \;=\; \begin{array}{r|rll} & (ax_1^{t+6}\langle 1\rangle, ax_1^{t+5}\langle 3\rangle) = & & \\ & 00 & 01 & \cdots \\ \hline (ax_1^{t+6}\langle 2\rangle, ax_1^{t+5}\langle 4\rangle) = 00 & 1 & 0 & \cdots \\ 01 & 0 & 0 & \cdots \\ 10 & 0 & 0 & \cdots \\ 11 & 0 & 1 & \cdots \end{array} \tag{84}$$

$$\rightarrow \;\; \sigma_X(1)^{n(0)} I_2^{\otimes 2} \; . \tag{85}$$

2. Node $(\underline{ax}_3^{t+6}\langle 2\rangle, \underline{bx}_2^{t+5}\langle 4\rangle, \underline{bx}_1^{t+5}\langle 4\rangle)$. One has

$$\begin{array}{l} A(ax_3^{t+6}\langle 2\rangle, bx_2^{t+5}\langle 4\rangle, bx_1^{t+5}\langle 4\rangle | ax_3^{t+6}\langle 1\rangle = 0, bx_2^{t+5}\langle 3\rangle, bx_1^{t+5}\langle 3\rangle) = \\ \sqrt{P_{\underline{x}_3|\underline{x}_2,\underline{x}_1}(ax_3^{t+6}\langle 2\rangle | bx_2^{t+5}\langle 3\rangle, bx_1^{t+5}\langle 3\rangle} \;\; \delta^{bx_2^{t+5}\langle 4\rangle}_{bx_2^{t+5}\langle 3\rangle} \delta^{bx_1^{t+5}\langle 4\rangle}_{bx_1^{t+5}\langle 3\rangle} \end{array} \quad . \tag{86}$$

If we indicate non-zero entries by a plus sign,

$$A \;=\; \begin{array}{r|lllll} & 000 & 001 & 010 & 011 & \cdots \\ \hline (ax_3^{t+6}, bx_2^{t+5}, bx_1^{t+5}) = 000 & + & & & & \cdots \\ 001 & & + & & & \cdots \\ 010 & & & + & & \cdots \\ 011 & & & & + & \cdots \\ 100 & + & & & & \cdots \\ 101 & & + & & & \cdots \\ 110 & & & + & & \cdots \\ 111 & & & & + & \cdots \end{array} \tag{87}$$

$$\rightarrow \;\; \sum_{\vec{b}\in Bool^2} e^{i\theta_{\vec{b}}\sigma_Y} \otimes P_{\vec{b}} \tag{88}$$

$$= \;\; \sum_{\vec{b}\in Bool^2} e^{i\theta_{\vec{b}}\sigma_Y(2)} P_{\vec{b}}(1,0) \; , \tag{89}$$

for some $\theta_{\vec{b}} \in \mathbb{R}$. This choice of A can be compiled using multiplexor methods.

6.2 Metropolis-Hastings Sampling

In this section, we will propose a method for doing Metropolis-Hastings sampling of a CB net on a *quantum* computer. The traditional method for doing Metropolis-Hastings sampling of a CB net on a *classical* computer is reviewed in Appendix B.2.

Compare Eq.(109) for the Gibbs algorithm with Eq.(122) for the Metropolis-Hastings algorithm. From this comparison we conclude that if in the Gibbs algorithm of Fig.5, we replace $P(x_i^{t+1}|(x^t)_{MB(i)})$ for $i \in E^c$ by the following, we will be doing Metropolis-Hastings.

$$P(x_i^{t+1}|(x^t)_{MB(i)}) \rightarrow \overline{Q}_i(x_i^{t+1}|x^t) + \delta_{x_i^{t+1}}^{x_i^t}[1 - \sum_{y_i} \overline{Q}_i(y_i|x^t)] \tag{90}$$

$$= \left\{ \begin{array}{l} \Theta(x_i^t \neq x_i^{t+1})\overline{Q}_i(x_i^{t+1}|x^t) \\ + \\ \Theta(x_i^t = x_i^{t+1})[1 - \sum_{y_i: y_i \neq x_i^t} \overline{Q}_i(y_i|x^t)] \end{array} \right\} . \tag{91}$$

A Appendix: Importance Sampling of CB Nets on a Classical Computer

In this Appendix, we review the importance sampling algorithm for CB nets on a classical computer.

Consider a CB net whose nodes are labeled in topological order by $(\underline{x}_1, \underline{x}_2, \ldots \underline{x}_{N_{nds}}) \equiv \underline{x}$. Assume that E (evidence set) and H (hypotheses set) are disjoint subsets of $Z_{1,N_{nds}}$, with $Z_{1,N_{nds}} - E \cup H$ not necessarily empty. Let $X^c = Z_{1,N_{nds}} - X$ for any $X \subset Z_{1,N_{nds}}$. Assume that we are given the prior evidence $(x)_E$, and the number of samples N_{sam} that we intend to collect.

Suppose x' is an arbitrary point in $val(\underline{x})$. (We'll use the unprimed x, as in $(x)_E$, to denote the evidence.) The probability matrices associated with each node of our CB net will be denoted by $P(x_i'|(x')_{pa(i)})$ for each $i \in Z_{1,N_{nds}}$. In addition, we will assume we are given **sampling probability matrices**, associated with each node of our CB net, denoted by $Q(x_i'|(x')_{pa(i)})$ for each $i \in Z_{1,N_{nds}}$. In all cases, these sampling matrices are constrained to satisfy

$$Q(x_i'|(x')_{pa(i)}) = P(x_i'|(x')_{pa(i)}) \ \ \forall i \in E^c \ . \tag{92}$$

Two important special cases of importance sampling are rejection sampling and likelihood weighted sampling. For **rejection sampling (RS)**,

$$Q(x_i'|(x')_{pa(i)}) = P(x_i'|(x')_{pa(i)}) \ \ \forall i \in Z_{1,N_{nds}} \ . \tag{93}$$

Hence, $Q(x') = P(x')$ for rejection sampling. For **likelihood weighted sampling**

(LWS) (a.k.a. likelihood weighting),

$$Q(x'_i|(x')_{pa(i)}) = \begin{cases} P(x'_i|(x')_{pa(i)}) & \forall i \in E^c \\ \delta(x_i, x'_i) & \forall i \in E \end{cases} . \tag{94}$$

Hence, $Q(x') = \delta_{(x)_E}^{(x')_E} \prod_{i \in E^c} P(x'_i|(x')_{pa(i)})$ for likelihood weighted sampling.

Under these assumptions, the importance sampling algorithm is given by Fig.9 (expressed in pseudo-code, pidgin C language).

```
For all (x)_H {W[(x)_H] = 0;}
W_tot = 0;
For samples k = 1, 2, ..., N_sam{
    L = 1;
    For nodes i = 1, 2, ..., N_nds{
        Generate x_i^(k) from Q(x_i|(x^(k))_pa(i));
        //Here, for LWS, x_i^(k) == x_i when i ∈ E.
        //pa(i) ⊂ Z_{1,i-1} so (x^(k))_pa(i) known at this point.
        if i ∈ E{
            if x_i^(k) == x_i{
                L *= P(x_i|(x^(k))_pa(i)) / Q(x_i|(x^(k))_pa(i));
                //Here P/Q = 1 for RS and P/Q = P for LWS.
            }else{//LWS never enters here
                go to next k;
            }
        }
    }//i loop (nodes)
    W[(x^(k))_H] += L;
    W_tot += L;
}//k loop (samples)
For all (x)_H {P((x)_H|(x)_E) = W[(x)_H] / W_tot;}
```

Figure 9: Algorithm for importance sampling of CB net on classical computer.

Claim 4 *For the algorithm of Fig.9,* $\frac{W[(x)_H]}{W_{tot}} \to P((x)_H|(x)_E)$ *as* $N_{sam} \to \infty$.

proof:

Define the likelihood ratio function:

$$L_E(x') = \prod_{i \in E} \frac{P(x'_i|(x')_{pa(i)})}{Q(x'_i|(x')_{pa(i)})} \tag{95}$$

for all $x' \in val(\underline{x})$. Clearly,

$$Q(x')L_E(x') = \prod_{i\in E^c}\{Q(x'_i|(x')_{pa(i)})\}\prod_{i\in E}\{P(x'_i|(x')_{pa(i)})\} \quad (96)$$

$$= P(x') \; . \quad (97)$$

For any function $g : val(\underline{x}) \to \mathbb{R}$, as $N_{sam} \to \infty$, the sample average $\overline{g(x^{(k)})}$ tends to:

$$\overline{g(x^{(k)})} = \frac{1}{N_{sam}}\sum_k g(x^{(k)}) \to \sum_{x'} Q(x')\delta[(x)_E,(x')_E]g(x') \; . \quad (98)$$

Therefore,

$$\frac{W[(x)_H]}{W_{tot}} = \frac{\frac{1}{N_{sam}}\sum_k L_E(x^{(k)})\delta[(x)_H,(x^{(k)})_H]}{\frac{1}{N_{sam}}\sum_k L_E(x^{(k)})} \quad (99)$$

$$\to \frac{\sum_{x'} P(x')\delta[(x)_{E\cup H},(x')_{E\cup H}]}{\sum_{x'} P(x')\delta[(x)_E,(x')_E]} \quad (100)$$

$$\to \frac{P((x)_{E\cup H})}{P((x)_E)} \; . \quad (101)$$

QED

B Appendix: Markov Chain Monte Carlo for CB Nets on a Classical Computer

In this Appendix, we review two examples (Gibbs and Metropolis-Hastings) of Markov Chain Monte Carlo (MCMC) sampling algorithms for CB nets on a classical computer.

A **Markov chain** is a CB net of the form

$$\underline{x}^0 \to \underline{x}^1 \to \underline{x}^2 \to \ldots \to \underline{x}^T \; , \quad (102)$$

where $val(\underline{x}^t)$ is independent of $t \in Z_{0,T}$.

It's clear from its graph that a Markov chain satisfies

$$P(x^{t+1}|x^t, x^{t-1}, \ldots x^0) = P(x^{t+1}|x^t) \; , \quad (103)$$

i.e, the probability that $\underline{x}^{t+1} = x^{t+1}$ at time $t+1$ is independent of what happened at all previous times except at the immediate past t. The $N_{\underline{x}^t} \times N_{\underline{x}^t}$ matrix with entries $P(\underline{x}^{t+1} = x|\underline{x}^t = x')$ is called the **transition matrix** of the Markov chain; we will represent it by $\mathcal{T}$. We will assume that $\mathcal{T}$ is independent of t (this property of $\mathcal{T}$ is called time invariance or time homogeneity).

Let $\pi : val(\underline{x}^t) \to \mathbb{R}$ be a probability vector ($\pi(x) \geq 0$ for all $x \in val(\underline{x}^t)$ and $\sum_x \pi(x) = 1$).

We say π is a **stationary distribution** of $\mathcal{T}$ if

$$\mathcal{T}\pi = \pi \ , \tag{104}$$

i.e., π is an eigenvector of $\mathcal{T}$ with unit eigenvalue.

We say π is a **detailed balance** of $\mathcal{T}$ if $\mathcal{T}(x'|x)\pi(x)$ is invariant under the exchange of x and x'; that is,

$$\mathcal{T}(x'|x)\pi(x) = \mathcal{T}(x|x')\pi(x') \ , \tag{105}$$

for all $x, x' \in val(\underline{x}^t)$. Detailed balance is tantamount to equilibrium since $\mathcal{T}(x'|x)\pi(x)$ is the probability flux being transmitted from state $\underline{x}^t = x$ to state $\underline{x}^{t+1} = x'$ after a long time, and $\mathcal{T}(x|x')\pi(x')$ is that being transmitted in the opposite direction, and these two are equal. Hence, it is not surprising that if π is a detailed balance of $\mathcal{T}$, then π is a stationary distribution of $\mathcal{T}$. Indeed, summing over x both sides of Eq.(105) proves this.

A **Markov Chain Monte Carlo (MCMC) sampling algorithm** is a method whereby, given a Markov chain $\underline{x}^0 \to \underline{x}^1 \to \underline{x}^2 \to \ldots$, we find a set of points in $val(\underline{x}^t)$ that is distributed according to the stationary distribution π of the Markov chain. π is taken to be the full probability distribution of a CB net. Next we discuss two examples of MCMC sampling algorithms: the Gibbs and the Metropolis-Hastings sampling algorithms. Actually, the Gibbs algorithm is a special case of the Metropolis-Hastings one, but I think it is pedagogically beneficial to discuss the Gibbs algorithm first, separately.

B.1 Appendix: Gibbs Sampling of CB Nets on a Classical Computer

In this Appendix, we review the Gibbs sampling algorithm for CB nets on a classical computer.

Consider a Markov chain $\underline{x}^0 \to \underline{x}^1 \to \underline{x}^2 \ldots \to \underline{x}^T$. Let $\underline{x}^t = (\underline{x}^t_1, \underline{x}^t_2, \ldots, \underline{x}^t_{N_{nds}})$ for each time t represent a separate copy of a CB net with nodes $\underline{x}^t_1, \underline{x}^t_2, \ldots, \underline{x}^t_{N_{nds}}$, and probability matrices $P(x^t_i|(x^t)_{pa(i)})$. (The nodes of the CB net $\underline{x}^t$ are not necessarily in topological order.) Assume that E (evidence set) and H (hypotheses set) are disjoint subsets of $Z_{1,N_{nds}}$, with $Z_{1,N_{nds}} - E \cup H$ not necessarily empty. Let $X^c = Z_{1,N_{nds}} - X$ for any $X \subset Z_{1,N_{nds}}$. Assume that we are given the prior evidence $(x)_E$. All probabilities in this section about the Gibbs algorithm will be conditioned implicitly on $(\underline{x}^t)_E = (x)_E$ for all t. The Gibbs algorithm is designed to respect this constraint, by never changing the value of $(\underline{x}^t)_E$ after it is initially set. Assume that we are given the last time T of the Markov chain, and the burn time t_{burn} ($0 << t_{burn} << T$).

Under these assumptions, the Gibbs sampling algorithm is given by Fig.10 (expressed in pseudo-code, pidgin C language).

For all $(x)_H$ $\{W[(x)_H] = 0;\}$
$W_{tot} = 0$;
Initialize $\underline{x}^0$ to some value x^0, subject to $(x^0)_E = (x)_E$;
For times $t = 0, 1, 2, \ldots, T-1\{$
    Draw i uniformly from $Z_{1,N_{nds}}$;
    if $i \in E\{$
        $x_i^{t+1} = x_i$;
    }else{
        Generate $x_i^{t+1} \sim P(x_i^{t+1}|(x^t)_{MB(i)})$;
    }
    $(x^{t+1})_{\{i\}^c} = (x^t)_{\{i\}^c}$;
    if $t > t_{burn}\{// \ 0 << t_{burn} << T$
        $W[(x^{t+1})_H] + +$;
        $W_{tot} + +$;
    }
$\}//t$ loop (times)
For all $(x)_H$ $\{P((x)_H|(x)_E) = \frac{W[(x)_H]}{W_{tot}};\}$

Figure 10: Algorithm for Gibbs sampling of CB net on classical computer. Nodes of CB net $\underline{x}^t$ are visited at random.

Claim 5 *For the algorithm of Fig.10, $P(\underline{x}^t = x^t)$ is a stationary distribution of $P(\underline{x}^{t+1} = x^{t+1}|\underline{x}^t = x^t)$. In other words,*

$$\sum_{x^t \in val(\underline{x}^t)} P(x^{t+1}|x^t)P(x^t) = P(x^{t+1}) \tag{106}$$

for all $x^{t+1} \in val(\underline{x}^t)$.

proof:

One begins by conditioning the transition matrix on the node index i:

$$P(x^{t+1}|x^t) = \frac{1}{N_{nds}} \sum_i P(x^{t+1}|x^t, i) \ . \tag{107}$$

Rather than proving Eq.(106), we will prove the stronger statement

$$\sum_{x^t} P(x^{t+1}|x^t, i)P(x^t) = P(x^{t+1}) \tag{108}$$

for all $i \in Z_{1,N_{nds}}$. If $P(x^{t+1})$ is a stationary distribution of $P(x^{t+1}|x^t, i) = \mathcal{T}(i)$ for any i, then it is a stationary distribution of any product $\mathcal{T}(i_1)\mathcal{T}(i_2)\cdots\mathcal{T}(i_n)$, for any

sequence $i_1, i_2, \ldots, i_n$ of i's.

Studying the algorithm of Fig.10 carefully, we conclude that

$$P(x^{t+1}|x^t, i) = \left[\Theta(i \in E^c)P(x_i^{t+1}|(x^t)_{MB(i)}) + \Theta(i \in E)\delta_{x_i^t}^{x_i^{t+1}}\right] \delta_{(x^t)_{\{i\}^c}}^{(x^{t+1})_{\{i\}^c}} \ . \quad (109)$$

For $i \in E$, $P(x^{t+1}|x^t, i) = \delta_{x^t}^{x^{t+1}}$, so Eq.(108) is clearly satisfied. For $i \in E^c$,

$$\begin{aligned}
\sum_{x^t} P(x^{t+1}|x^t, i)P(x^t) &= \sum_{x_i^t, (x^t)_{\{i\}^c}} P(x_i^{t+1}|(x^t)_{\{i\}^c})\delta_{(x^t)_{\{i\}^c}}^{(x^{t+1})_{\{i\}^c}} P(x^t) && (110)\\
&= \sum_{x_i^t} P(x_i^{t+1}|(x^{t+1})_{\{i\}^c})P((x^{t+1})_{\{i\}^c}, x_i^t) && (111)\\
&= P(x_i^{t+1}|(x^{t+1})_{\{i\}^c})P((x^{t+1})_{\{i\}^c}) && (112)\\
&= P(x^{t+1}) \ . && (113)
\end{aligned}$$

QED

Rather than choosing nodes of the graph $\underline{x}^t$ at random, one can sweep through all of them, in a fixed deterministic order, repeating this all-nodes-sweep N_{gra} times. Hence, we can replace the algorithm of Fig.10 by the one of Fig.11.

The sample of points in $val(\underline{x}^t)$ generated by this "nodes in fixed order" algorithm isn't time invariant for t differences $\Delta t = 1$, but is time invariant for $\Delta t = N_{nds}$.

B.2 Appendix: Metropolis-Hastings Sampling of CB Nets on a Classical Computer

In this Appendix, we review the Metropolis-Hastings sampling algorithm for CB nets on a classical computer.

Consider a Markov chain $\underline{x}^0 \to \underline{x}_1 \to \underline{x}^2 \ldots \to \underline{x}^T$. Let $\underline{x}^t = (\underline{x}_1^t, \underline{x}_2^t, \ldots, \underline{x}_{N_{nds}}^t)$ for each time t represent a separate copy of a CB net with nodes $\underline{x}_1^t, \underline{x}_2^t, \ldots, \underline{x}_{N_{nds}}^t$. and probability matrices $P(x_i^t|(x^t)_{pa(i)})$. (The nodes of the CB net $\underline{x}^t$ are not necessarily in topological order.) Assume that E (evidence set) and H (hypotheses set) are disjoint subsets of $Z_{1,N_{nds}}$, with $Z_{1,N_{nds}} - E \cup H$ not necessarily empty. Let $X^c = Z_{1,N_{nds}} - X$ for any $X \subset Z_{1,N_{nds}}$. Assume that we are given the prior evidence $(x)_E$. All probabilities in this section about the Metropolis-Hastings algorithm will be conditioned implicitly on $(\underline{x}^t)_E = (x)_E$ for all t. The Metropolis-Hastings algorithm is designed to respect this constraint, by never changing the value of $(\underline{x}^t)_E$ after it is initially set. Assume that we are given the last time T of the Markov chain, the burn time t_{burn} ($0 << t_{burn} << T$), and **sampling probability distributions** $Q_i(y_i|x_i^t, (x^t)_{MB(i)})$ where $i \in Z_{1,N_{nds}}$, $y_i \in val(\underline{x}_i^t)$, $x^t \in val(\underline{x}^t)$.

Under these assumptions, the Metropolis-Hastings sampling algorithm is given by Fig.12 (expressed in pseudo-code, pidgin C language).

For all $(x)_H$ $\{W[(x)_H] = 0;\}$
$W_{tot} = 0$;
Initialize $\underline{x}^0$ to some value x^0, subject to $(x^0)_E = (x)_E$;
$t = 0$;
For graphs $g = 1, 2, 3, \ldots, N_{gra}\{$
    For nodes $i = 1, 2, \ldots, N_{nds}\{$;
        if $i \in E\{$
            $x_i^{t+1} = x_i$;
        }else{
            Generate $x_i^{t+1} \sim P(x_i^{t+1}|(x^t)_{MB(i)})$;
        }
        $(x^{t+1})_{\{i\}^c} = (x^t)_{\{i\}^c}$;
        $t++$;
    }//i loop (nodes)
    if $t > t_{burn}\{$// $0 << t_{burn} << N_{nds}N_{gra}$
        $W[(x^t)_H]++$;
        $W_{tot}++$;
    }
}//g loop (graphs)
For all $(x)_H$ $\{P((x)_H|(x)_E) = \frac{W[(x)_H]}{W_{tot}};\}$

Figure 11: Algorithm for Gibbs sampling of CB net on classical computer. Nodes of CB net $\underline{x}^t$ are visited in a fixed deterministic order.

Claim 6 *For the algorithm of Fig.12,* $P(\underline{x}^t = x^t)$ *is a stationary distribution of* $P(\underline{x}^{t+1} = x^{t+1}|\underline{x}^t = x^t)$. *In other words,*

$$\sum_{x^t \in val(\underline{x}^t)} P(x^{t+1}|x^t)P(x^t) = P(x^{t+1}) \tag{114}$$

for all $x^{t+1} \in val(\underline{x}^t)$.

proof:

One begins by conditioning the transition matrix on the node index i:

$$P(x^{t+1}|x^t) = \frac{1}{N_{nds}} \sum_i P(x^{t+1}|x^t, i) \; . \tag{115}$$

Rather than proving Eq.(114), we will prove the stronger statement

$$\sum_{x^t} P(x^{t+1}|x^t, i)P(x^t) = P(x^{t+1}) \tag{116}$$

For all $(x)_H$ $\{W[(x)_H] = 0;\}$
$W_{tot} = 0$;
Initialize $\underline{x}^0$ to some value x^0, subject to $(x^0)_E = (x)_E$;
For times $t = 0, 1, 2, \ldots, T-1\{$
    Draw i uniformly from $Z_{1,N_{nds}}$;
    if $i \in E\{$
        $x_i^{t+1} = x_i$;
    $\}$else$\{$
        Generate $y_i \sim Q_i(y_i|x_i^t, (x^t)_{MB(i)})$;
        Draw u_i uniformly from the interval $[0, 1]$;
        $\alpha_i = \min\left\{1, \frac{Q_i(x_i^t|y_i,(x^t)_{MB(i)})P(y_i|(x^t)_{MB(i)})}{Q_i(y_i|x_i^t,(x^t)_{MB(i)})P(x_i^t|(x^t)_{MB(i)})}\right\}$;
        if $(u_i < \alpha_i)\{x_i^{t+1} = y_i;\}$ else $\{x_i^{t+1} = x_i^t;\}$
        //if $Q_i = P$, then $\alpha_i = \min(1, 1) = 1$, and get Gibbs
    $\}$
    $(x^{t+1})_{\{i\}^c} = (x^t)_{\{i\}^c}$;
    if $t > t_{burn}\{$// $0 << t_{burn} << T$
        $W[(x^{t+1})_H] + +$;
        $W_{tot} + +$;
    $\}$
$\}$//t loop (times)
For all $(x)_H$ $\{P((x)_H|(x)_E) = \frac{W[(x)_H]}{W_{tot}};\}$

Figure 12: Algorithm for Metropolis-Hastings sampling of CB net on classical computer.

for all $i \in Z_{1,N_{nds}}$. If $P(x^{t+1})$ is a stationary distribution of $P(x^{t+1}|x^t, i) = \mathcal{T}(i)$ for any i, then it is a stationary distribution of any product $\mathcal{T}(i_1)\mathcal{T}(i_2)\cdots\mathcal{T}(i_n)$, for any sequence $i_1, i_2, \ldots, i_n$ of i's.

Studying the algorithm of Fig.12 carefully, we conclude the following. For $i \in E$, $P(x^{t+1}|x^t, i) = \delta_{x^t}^{x^{t+1}}$, so Eq.(116) is clearly satisfied. For $i \in E^c$,

$$P(x^{t+1}|x^t, i) = \sum_{y_i} \int_0^1 du_i P(x^{t+1}|x^t, y_i, u_i, i) P(y_i|x^t, i) P(u_i|i) \; . \tag{117}$$

Eq.(117) comes from the CB net of Fig.13. The 3 probabilities occurring on the right hand side of Eq.(117) are given by

$$P(x^{t+1}|x^t, y_i, u_i, i) = \delta_{(x^{t+1})_{\{i\}^c}}^{(x^t)_{\{i\}^c}} \;\; \delta_{x_i^{t+1}}^{y_i\Theta(u_i<\alpha_i)+x_i^t\Theta(u_i>\alpha_i)} \; , \tag{118a}$$

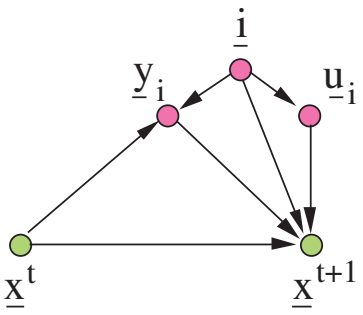


Figure 13: CB net connecting the random variables used in the Metropolis-Hastings algorithm.

$$P(y_i|x^t, i) = Q_i(y_i|x_i^t, (x^t)_{MB(i)}) \ , \tag{118b}$$

and

$$P(u_i|i) = 1 \ . \tag{118c}$$

(u_i is a continuous random variable.)

One can "sum over" the node u_i of Fig.13:

$$\int_0^1 du_i P(x^{t+1}|x^t, y_i, u_i, i) = \delta^{(x^t)_{\{i\}^c}}_{(x^{t+1})_{\{i\}^c}} \left[\delta^{y_i}_{x_i^{t+1}}\alpha_i + \delta^{x_i^t}_{x_i^{t+1}}(1-\alpha_i)\right] \ . \tag{119}$$

Define

$$\overline{Q}_i(y_i|x^t) \equiv \alpha_i Q_i(y_i|x_i^t, (x^t)_{MB(i)}) \ . \tag{120}$$

One can also sum over the node y_i of Fig.13:

$$P(x^{t+1}|x^t, i) = \sum_{y_i} \delta^{(x^t)_{\{i\}^c}}_{(x^{t+1})_{\{i\}^c}} \left[\delta^{y_i}_{x_i^{t+1}}\alpha_i + \delta^{x_i^t}_{x_i^{t+1}}(1-\alpha_i)\right] Q_i(y_i|x_i^t, (x^t)_{MB(i)}) \tag{121}$$

$$= \delta^{(x^t)_{\{i\}^c}}_{(x^{t+1})_{\{i\}^c}} \left\{ \overline{Q}_i(x_i^{t+1}|x^t) + \delta^{x_i^t}_{x_i^{t+1}}[1 - \sum_{y_i} \overline{Q}_i(y_i|x^t)] \right\} \ . \tag{122}$$

Note that

$$P(x^t) = P(x_i^t|(x^t)_{\{i\}^c}) P((x^t)_{\{i\}^c}) \tag{123}$$

$$= P(x_i^t|(x^t)_{MB(i)}) P((x^t)_{\{i\}^c}) \ . \tag{124}$$

Hence,

$$\begin{aligned}
\overline{Q}(y_i|x^t)P(x^t) &= \min\left\{1, \frac{Q_i(x_i^t|y_i,(x^t)_{MB(i)})P(y_i|(x^t)_{MB(i)})}{Q_i(y_i|x_i^t,(x^t)_{MB(i)})P(x_i^t|(x^t)_{MB(i)})}\right\} Q_i(y_i|x_i^t,(x^t)_{MB(i)})P(x^t) \\
&= \min\left\{\begin{array}{l} Q_i(y_i|x_i^t,(x^t)_{MB(i)})P(x_i^t|(x^t)_{MB(i)}), \\ Q_i(x_i^t|y_i,(x^t)_{MB(i)})P(y_i|(x^t)_{MB(i)}) \end{array}\right\} P((x^t)_{\{i\}^c}) \, . \qquad (125)
\end{aligned}$$

From Eq.(125), we see that $\overline{Q}(y_i|x^t)P(x^t)$ is invariant under the exchange of y_i and x_i^t. In other words, $P(x^t)$ is a detailed balance of $\overline{Q}(y_i|x^t)$ under the exchange of y_i and x_i^t. However, $\sum_{y_i} \overline{Q}(y_i|x^t) \neq 1$, so $\overline{Q}(y_i|x^t)$ is not a probability distribution in y_i.

Now we have

$$\begin{aligned}
\sum_{x^t} P(x^{t+1}|x^t,i)P(x^t) &= \sum_{x^t} \delta^{(x^t)_{\{i\}^c}}_{(x^{t+1})_{\{i\}^c}} \left\{\begin{array}{c} \overline{Q}_i(x_i^{t+1}|x^t) \\ + \\ \delta^{x_i^t}_{x_i^{t+1}}[1 - \sum_{y_i} \overline{Q}_i(y_i|x^t)] \end{array}\right\} P(x^t) \\
&= T_1 + T_2 + T_3 \, , \qquad (126)
\end{aligned}$$

where

$$\begin{aligned}
T_1 &= \sum_{x^t} \delta^{(x^t)_{\{i\}^c}}_{(x^{t+1})_{\{i\}^c}} \overline{Q}_i(x_i^{t+1}|x^t)P(x^t) \qquad (127) \\
&= \sum_{x_i^t} \overline{Q}_i(x_i^{t+1}|x_i^t,(x^{t+1})_{\{i\}^c})P(x_i^t,(x^{t+1})_{\{i\}^c}) \qquad (128) \\
&= \sum_{x_i^t} \overline{Q}_i(x_i^t|x_i^{t+1},(x^{t+1})_{\{i\}^c})P(x_i^{t+1},(x^{t+1})_{\{i\}^c}) \qquad (129) \\
&= \sum_{x_i^t} \overline{Q}_i(x_i^t|x^{t+1})P(x^{t+1}) \, , \qquad (130)
\end{aligned}$$

(To go from Eq.(128) to Eq.(129), we used the fact that $P(x^t)$ is a detailed balance of $\overline{Q}(y_i|x^t)$.)

$$T_2 = \sum_{x^t} \delta^{x^t}_{x^{t+1}} P(x^t) = P(x^{t+1}) \, , \qquad (131)$$

and

$$\begin{aligned}
T_3 &= -\sum_{x^t} \delta^{x^t}_{x^{t+1}} \sum_{y_i} \overline{Q}_i(y_i|x^t)P(x^t) \qquad (132) \\
&= -\sum_{y_i} \overline{Q}_i(y_i|x^{t+1})P(x^{t+1}) \qquad (133) \\
&= -T_1 \, . \qquad (134)
\end{aligned}$$

QED

As we pointed out for the Gibbs algorithm in Appendix B.1, rather than choosing nodes of the graph $\underline{x}^t$ at random, one can sweep through all the nodes of the graph in a fixed order, repeating this sweep N_{gra} times. Hence, we can replace the algorithm of Fig.12 by an alternative one. We leave the details to the reader.

There are several important special cases of the algorithm of Fig.12:

1. When $Q_i(y_i|x_i^t, (x^t)_{MB(i)}) = P(y_i|(x^t)_{MB(i)})$, we get $\alpha_i = \min(1, 1) = 1$, yielding the Gibbs algorithm discussed in Appendix B.1.

2. When $N_{nds} = 1$, there is no need for i subscripts in x_i^t or y_i. There is also no possibility of evidence. Hence, the algorithm of Fig.12 simplifies to the one in Fig.14.

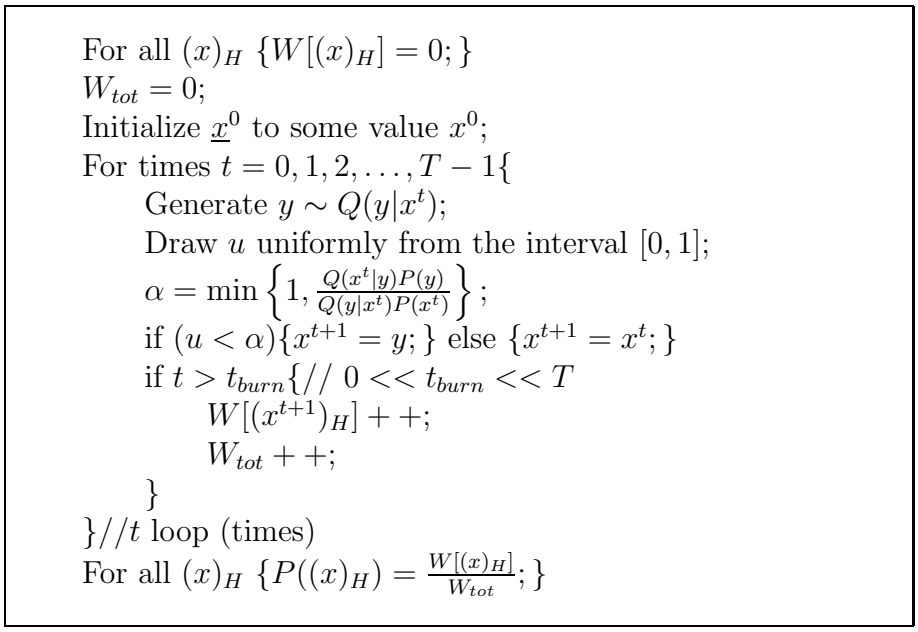


Figure 14: Algorithm for Metropolis-Hastings sampling of CB net on classical computer. Special case where $N_{nds} = 1$.

3. If $Q_i(x_i^t|y_i, (x^t)_{MB(i)})$ is invariant under the exchange of x_i^t and y_i, then

$$\alpha_i = \min\left\{1, \frac{P(y_i|(x^t)_{MB(i)})}{P(x_i^t|(x^t)_{MB(i)})}\right\} . \tag{135}$$

In particular, if $N_{nds} = 1$,

$$\alpha = \min\left\{1, \frac{P(y)}{P(x^t)}\right\} . \tag{136}$$

This is called the **Metropolis sampling algorithm**. It was invented prior to the Metropolis-Hastings one.

The algorithm of Fig.12 and the proof of Claim 6 are both fairly complicated. One wonders, why do they work? What logic motivated the invention of the algorithm? Here is an intuitive explanation of what is going on. Define

$$\mathcal{F}(x_i^t \leftarrow y_i) \equiv Q_i(x_i^t|y_i, (x^t)_{MB(i)})P(y_i|(x^t)_{MB(i)}) \ . \tag{137}$$

Define $\mathcal{F}(y_i \leftarrow x_i^t)$ as the expression on the right hand side of Eq.(137), but x_i^t and y_i exchanged. $\mathcal{F}(x_i^t \leftarrow y_i)$ is the probability flux flowing from state y_i to state x_i^t, and $\mathcal{F}(y_i \leftarrow x_i^t)$ is the flux in the opposite direction. Now α_i, often called the "acceptance probability", can be expressed as

$$\alpha_i = \min\left\{1, \frac{\mathcal{F}(x_i^t \leftarrow y_i)}{\mathcal{F}(y_i \leftarrow x_i^t)}\right\} \ . \tag{138}$$

Note that this definition of α_i puts it in the interval $[0, 1]$, as required for a probability. Recall that the algorithm defines:

$$x_i^{t+1} = y_i\Theta(u_i < \alpha_i) + x_i^t\Theta(u_i > \alpha_i) \ . \tag{139}$$

Thus,

1. If $\mathcal{F}(x_i^t \leftarrow y_i) << \mathcal{F}(y_i \leftarrow x_i^t)$, then $\alpha_i = \mathcal{F}(x_i^t \leftarrow y_i)/\mathcal{F}(y_i \leftarrow x_i^t) << 1$. In this case, we
   (a) Set $x_i^{t+1} = y_i$ (i.e., accept the new value), doing this infrequently, α_i of the time.
   (b) Set $x_i^{t+1} = x_i^t$ (i.e., keep the old value), doing this frequently, $1 - \alpha_i$ of the time.
2. If $\mathcal{F}(x_i^t \leftarrow y_i) > \mathcal{F}(y_i \leftarrow x_i^t)$, then $\alpha_i = 1$. In this case, we always set $x_i^{t+1} = y_i$ (i.e., accept the new value).

Most of the time (except in case 1a), we are trying to "buck (counteract) the trend" that state x_i^t is either gaining or loosing weight. We don't buck the trend always, because we want to allow a small probability of escaping local minima.

C Appendix: Quantum Circuits for Two Examples of Ref.[4]

Figs.15 and 16 are quantum circuits for the two-body scattering and Asia nets, respectively, that are discussed in Ref.[4].

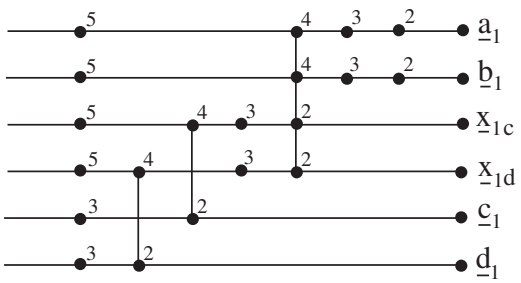


Figure 15: Quantum circuit for two body scattering QB net of Ref.[4]

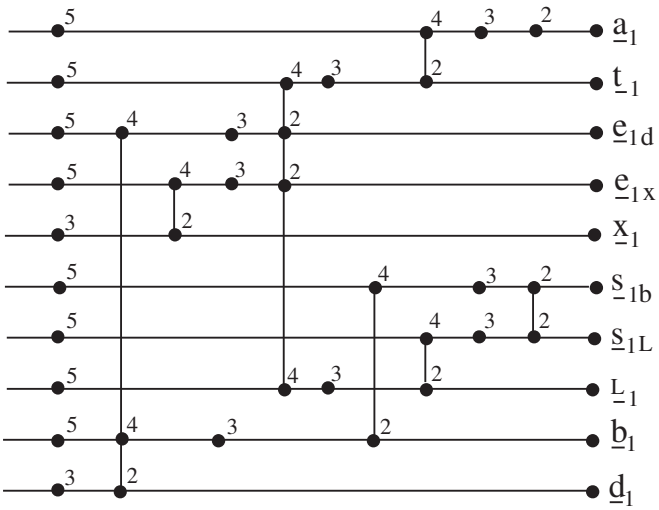


Figure 16: Quantum circuit for Asia QB net of Ref.[4]

References

[1] http://en.wikipedia.org/wiki/Inverse_transform_sampling

[2] http://en.wikipedia.org/wiki/Rejection_sampling

[3] M. Jordan (Editor) *Learning in Graphical Models* (1999, The MIT Press)

[4] R.R. Tucci, "Quantum Computer as a Probabilistic Inference Engine", arXiv:quant-ph/0004028 v2 .

[5] R.R. Tucci, "Use of a Quantum Computer and the Quick Medical Reference To Give an Approximate Diagnosis", arXiv:0806.3949

[6] Lov K. Grover, "Rapid sampling through quantum computing", quant-ph/9912001

[7] Pawel Wocjan, Anura Abeyesinghe, "Speed-up via Quantum Sampling", arXiv:0804.4259

[8] Pawel Wocjan, Chen-Fu Chiang, Anura Abeyesinghe, Daniel Nagaj, "Quantum Speed-up for Approximating Partition Functions", arXiv:0811.0596

[9] Charles Fox, "An Entangled Bayesian Gestalt: Mean-field, Monte-Carlo and Quantum Inference in Hierarchical Perception", DPhil thesis, Oxford, 2008

[10] R.R.Tucci, "QC Paulinesia", quant-ph/0407215

[11] G.H. Golub and C.F. Van Loan, *Matrix Computations, Third Edition* (John Hopkins Univ. Press, 1996).

[12] R.R. Tucci, "A Rudimentary Quantum Compiler(2cnd Ed.)", arXiv:quant-ph/9902062

[13] Ville Bergholm, Juha J. Vartiainen, Mikko Mottonen, Martti M. Salomaa. "Quantum circuits with uniformly controlled one-qubit gates", arXiv:quant-ph/0410066